DOCUMENT RESUME

ED 070 299 EM 010 592

AUTHOR	Rigney, Joseph W.; And Others
TITLE	Computer-Aided Performance Training for Diagnostic and Procedural Tasks.
INSTITUTION	University of Southern California, Los Angeles. Dept. of Psychology.
SPONS AGENCY	Office of Naval Research, Washington, D.C. Personnel and Training Research Programs Office.
REPORT NO	USC-TR-70
PUB DATE	Oct 72
NOTE	97p.; See Also EM 010 593
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Behavioral Science Research; *Computer Assisted Instruction; Computer Graphics; Computers; Instructional Technology; *Programed Instruction; *Programed Materials; Programing Languages; *Task Performance
IDENTIFIERS	LISP; TASKTEACH

ABSTRACT

Two computer programs for computer-assisted performance training were developed to give the students the opportunity for concentrated practice of troubleshooting and procedural tasks in naval electronics. In contrast to the usual approach taken in computer-assisted instruction (CAI), these programs simulate essential aspects of devices and tasks and continuously update their states during a practice session; they generate responses to a student's inputs from that student's history and simple list-structures; they are made specific to particular devices and tasks by data modules (therefore no "CAI language" is used); and they offer the student several options for drills and for receiving advice during practice. The LISP variation of the TASKTEACH programs was used and has been reviewed. These programs are being converted to run in a new type of program graphics terminal with two integral minicomputers. This terminal will be the basis for small "stand alone" CAI systems offering static and dynamic graphics, random-access photographic slides, and front panel simulators. (MC)

BEHAVIORAL TECHNOLOGY LABORATORIES

Department of Psychology
University of Southern California

DEPARTMENT OF PSYCHOLOGY

UNIVERSITY OF SOUTHERN CALIFORNIA

Technical Report No. 70

COMPUTER-AIDED PERFORMANCE TRAINING FOR DIAGNOSTIC AND PROCEDURAL TASKS

October 1972

Joseph W. Rigney
Douglas M. Towne
Carole A. King
Edward T. Langston

Prepared for

Personnel and Training Research Programs
Psychological Sciences Division
Office of Naval Research

Contract N00014-67-A-0269-0012
Contract Authority Identification No. NR 154-295

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified

1. ORIGINATING ACTIVITY (Corporate author) Behavioral Technology Laboratories University of Southern California Los Angeles, California 90007	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

COMPUTER-AIDED PERFORMANCE TRAINING FOR DIAGNOSTIC AND PROCEDURAL TASKS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report 70 October 1972

5. AUTHOR(S) (First name, middle initial, last name)

Joseph W. Rigney Carole A. King
Douglas M. Towne Edward T. Langston

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1972	82	9

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0269-0012 b. PROJECT NO NR154-295 c. d.	Technical Report 70 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC RELEASE AND SALE;
ITS DISTRIBUTION IS UNLIMITED

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Personnel and Training Research Programs Psychological Sciences Division Office of Naval Research

13. ABSTRACT

Two computer programs for computer-aided performance training were developed. These programs were designed to give students the opportunity for concentrated practice of troubleshooting and of procedural tasks. In contrast to the usual approach taken in CAI, these programs simulate essential aspects of devices and tasks and continuously update their states during a practice session; they generate responses to a student's inputs from that student's history and simple list-structures; they are made specific to particular devices and tasks by data modules, therefore no "CAI language" is used; and they offer the student several options for drills and for receiving advice during practice. (U)

These programs are being converted to run in a new type of programmable graphics terminal with two integral minicomputers. This terminal will be the basis for small "stand alone" CAI systems offering static and dynamic graphics, random-access photographic slides, and front panel simulators. (U)

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified

Security Classification

Computer-Aided Instruction						
Task Simulation						
Generative Programming						
Maintenance Training						
Computer Programming						

ACKNOWLEDGMENTS

This work was made possible through the support and encouragement of Dr. Marshall Farr and Dr. Joseph Young, Personnel and Training Research Programs, and Dr. Glenn Bryan, Director of Psychological Research Division, Office of Naval Research. Their continued support is gratefully acknowledged.

This U.S. Naval Schools Command, Treasure Island, San Francisco, supplied students for testing the program. The interest and encouragement of Mr. Robert Cushing, Educational Advisor at that Command, is most appreciated.

ABSTRACT

Two computer programs for computer-aided performance training were developed. These programs were designed to give students the opportunity for concentrated practice of troubleshooting and of procedural tasks. In contrast to the usual approach taken in CAI, these programs simulate essential aspects of devices and tasks and continuously update their states during a practice session; they generate responses to a student's inputs from that student's history and simple list-structures; they are made specific to particular devices and tasks by data modules, therefore no "CAI language" is used; and they offer the student several options for drills and for receiving advice during practice.

These programs are being converted to run in a new type of programmable graphics terminal with two integral minicomputers. This terminal will be the basis for small "stand alone" CAI systems offering static and dynamic graphics, random-access photographic slides, and front panel simulators.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

COMPUTER-AIDED PERFORMANCE TRAINING FOR DIAGNOSTIC AND PROCEDURAL TASKS

SECTION I. VARIETIES OF CAI

There are numerous ways to use computers in interaction with students to create and to control learning experiences. The way most often used evolved from programmed instruction, not surprisingly, since an obvious way to get started was to implement this known instructional procedure with relatively simple data-processing technology. In this approach, detailed organization of the material to be learned, and detailed specification of computer and student responses in the interaction falls on the person who prepares the course before instruction commences. The unit of interaction often is somewhat similar to an objective test item format and is called a frame. Hundreds of frames must be written by the course preparer and tried out on students, following a procedure not significantly different from preparation of programmed instruction. The prescriptions from programmed-instruction days regarding how to construct "good" frames are applicable. Eventually, a long sequence of tested frames is developed and judged suitable for use on a particular population of students.

Other varieties of CAI have been described by Carbonnel (1972), Feurtzig (1971), Uttal (1968), and others. In some of these, the investigator has been concerned with finding ways of using more of the potential of computers. After all, they are, in comparison to students, very high speed processors, capable of executing thousands or hundreds of

thousands of instructions in a fraction of a second. They can store millions of units of information, and access any selected group of units in milliseconds. With the proper logic, they can sort, select, transform, concatenate, substitute, replace, search, update, decode, or do any of dozens of other operations between the time a student enters data through a terminal and the time an "immediate" response is required.

Our long-term objective has been to find ways to utilize more of the power inherent in the digital computer for creating and controlling conditions for learning that would be impossible by any other means. The computer's unique role is to do things that could not otherwise be done at all and that immensely enhance the effectiveness of learning environments. We do not suppose that our techniques have fully explored this potential of the computer, but it is toward this end that we have produced the programs described here.

We do not believe our approach is the only approach, or that the evolution of CAI from programmed instruction is not fruitful. CAI is in very early stages of development. It is in these early stages, we do believe, that many different approaches should be developed and tried out.

The principal characteristics of our approach are:

1. The computer programs contain the majority of the logic for creating the dialogue with the student. There is no "CAI language" in the sense of Coursewriter, Planit, etc. The programs operate on simple lists which constitute the "data module."
2. The computer programs contain general logic. The addition of the data module makes a combination that is specific to a particular equipment upon which the tasks are to be performed.

3. The interaction with a student is unique to that student. The interaction is generated by reference to continuously updated histories of that student's responses and of task and equipment states.

4. There are no long sequences of programmed instruction frames to write. Instead, comparatively short list-structures are prepared from analysis of equipment and tasks.

5. This brand of CAI is intended to give students the opportunity to practice job-related tasks. It is not oriented toward teaching "theory."

Of the necessary hardware and software elements for a complete CAI system, we have chosen to work on the computer programs that sustain the interaction with a student while he is learning. It was convenient to use time-sharing systems to develop these programs. In fact, they could not have been developed without the powerful programming languages such as Lisp, PL/1, Basic, and Fortran, that these systems offer. These resources make writing and debugging programs on the order of ten times faster in comparison to using batch systems. It is not particularly convenient, and it is not yet cost-effective, to use these same time-sharing systems to implement CAI in a school. Therefore, this report will be about computer programs that have been developed and tested as programs. Implementation testing will follow, using quite different, and much cheaper hardware, than was used for program development.

SECTION II. TASK STRUCTURES

A convenient, and quite general way to describe procedural tasks is in terms of goal and action hierarchies. Some endpoint goal is identified. This can be attained only by successively attaining intermediate goals. Each intermediate goal is accomplished by performing all or a subset of a set of actions. Variations in sequence and other relationships among actions and goals that depart from a simple hierarchical structure can be described by clustering operators. These three elements, goal descriptions, action descriptions, and clustering operators, are necessary to describe any task structure. The task structure may be extended downward to a level of detail that suits the requirements for its use. When the description is to be used for generating instructions to a student telling him how to perform a task, the downward extension need go only as far as necessary to fully instruct the most naive students from a particular population. This usually is far above the "time-and-motion" level used by industrial engineers. Where tasks are performed on equipment interfaces, it is sufficient to specify what control is to be operated, and the terminal state of the control or associated display that is required as a consequence of the operation. Several advantages of this approach to describing task structures are:

1. Computer program logic can be developed to process the syntatic structure composed of goal, action, and clustering operator symbols, and to supply the semantic structure to the student by substituting verbal statements for these syntatic codes when communicating with the student.

2. The hierarchical nature of the description allows the student to work at any level of detail appropriate to his needs. If he already knows how to perform sections of a task structure, he can skip through these at higher goal levels, until he reaches a section he does not know so well. He can go into this section at a finer level of detail in his practice.

3. Actions are clustered under goals, so that the two form convenient units to learn and remember. Actions belonging to a cluster can be associated with the name, the goal description, of that cluster. Thus, there is an isomorphism between the task structure and organizations known to facilitate learning and remembering.

4. "Maps" of task structures can be used to help the student grasp the syntatic relationships in the structure.

5. Clustering operators permit descriptions to the computer program and to the student of special goal-action relationships encountered in many tasks. The operators that have been found sufficient so far are described below. The code names for them are recognized by the computer programs for teaching procedures: SEQ, ANY, ALL, ANYN, IFIN, IFEX, IFIC, REP, REPU, UNDO.

SEQ: All subgoals or actions associated with the goal must be performed in a prescribed sequence.

ANY: Performing any one of a set of goals or actions is sufficient to accomplish the next higher goal.

ANYN: Performing any prescribed subset (N) of a set of goals or actions is sufficient to progress in the task structure.

ALL: All goals or actions in a set must be performed, but they can be performed in any order. Sequence is unimportant.

IFIN: If some condition internal to the task structure is true, then some particular action should be taken, otherwise skip that action. For example, if the student has put a multi-mode device in a particular mode, there may be a goal or action peculiar to that mode. The procedural computer program decodes this operator and informs the student at the appropriate point in its interaction with him.

IFEX: Without on-line front-panel simulators, the computer program cannot sense the results of certain actions on displays. In general, with this operator, the program asks the student questions about the "state" of the external world. It uses his answers to select the immediately following part of the task structure.

IFIC: (If certain initial conditions are true..) many devices have attachments, and there may be a variety of other initial conditions from which the student wishes to establish a particular configuration for a practice session. This operator allows the computer program to ask the student questions about which of a set of initial conditions the student wishes to be true. This subset of initial conditions then is used to select appropriate sections of the task structure. For example, if there is no telephoto lens for a camera the student is learning to operate the part of the task structure concerned with using a telephoto lens is left out of the practice session.

REP: Certain goal-actions clusters are used repetitively. For example, removing an access plate may require removing ten screws. The actions for unscrewing a screw would be repeated nine times. This operator allows the program to handle this situation without requiring that the goal-actions cluster be represented in the task structure ten times.

REPU: Frequently, actions must be repeated an indefinite number of times until a desired result occurs. This operator represents this situation.

UNDO: In working on devices, there are occasions when a goal that was attained at one stage must be "undone." A common example is detuning a transmitter after it has been tuned. This operator allows the program to instruct the student properly from only one specification of the cluster.

DIFFICULTY LEVEL: Goal-actions clusters may be categorized in terms of difficulty level, if expert judgment is available, or if some optimizing algorithm was used. This operator should not be used without these sources of information. By setting a desired difficulty level, a student can learn to perform the task in a way which is suited to his level of learning. As his skill increases, and he requests higher levels of difficulty, the program will introduce new alternatives which are more difficult to perform. An alternative strategy was used in the testing of the procedural program. Students could repeat practicing parts of the task structure that were difficult for them as many times as necessary to achieve a criterion level.

SECTION III. EQUIPMENT STRUCTURES

It is reasonable to suppose that students learning to operate or to troubleshoot equipment develop some sort of mental representation of equipment structures that serves them, with appropriate "extended memories" in the form of visual aids and cues, and with appropriate feedback and knowledge of results, in guiding their performance. This mental representation probably includes a number of different ways of "looking at" structure, depending on the immediate performance requirements, made possible in part by the associative nature of human memory. A physical analog, no doubt immensely cruder, and probably far too deterministic, can be found in circuit analysis computer programs. For example, one widely used program can "look at" a circuit from the standpoint of DC voltages, AC rms voltages, effects of particular transients, sensitivity relationships, and worst case conditions.

If this assumption is correct, a primary objective of training must be to teach structure in ways such that the student is led to develop a sufficient number of related mental representations. Electronic equipment, e.g., a radar repeater, characteristically is described in technical manuals in highly fragmented ways that require the student to spend a great deal of time analyzing schematics and block diagrams to put together more useful mental representations. For example, if the goal is to learn to troubleshoot an electronic equipment from front panel symptoms, then the relationships among such symptoms and the circuitry must be learned. One way to facilitate this learning is to give the student "maps" of this

particular structure that he can refer to while practicing troubleshooting Once he learns effective troubleshooting procedures, he can, if necessary, learn to reorganize the more general maps in his technical manual. But, to ask him to mentally reorganize material in the technical manual while attempting to practice troubleshooting on-line with a CAI system would be intermixing two difficult tasks.

There are variously sophisticated ways of presenting equipment maps to students. In the present case, during the tests of the computer programs, the simplest possible way was used: printed diagrams with explanatory text in reference manuals. However, a longer-term objective would be to help the student develop what might be called "associative muiti-representations" (AMRS) of structures. That is, many related semantic structures could be associated with each other and could be derived from the same "bare bones" or syntactic structure. We might suppose that the result would be somewhat analogous to results of language learning. Some sort of "deep structure" might be learned that would serve as the basis for variations which would be useful, in combination with similar AMRS of task structures, in generating the performance required at the moment. This longer term objective might be approached through the use of computer graphics, a possibility that now is under investigation. It is expected that this will be an effective way to extend the current purely drill and practice CAI programs into instruction in "theory."

SECTION IV. INSTRUCTIONAL STRATEGY

Rigney, Fromer, and Bond (1967) discussed fundamental considerations for an instructional strategy for computer-aided performance training, and reviewed several adaptive control and optimization models with possible transfer value for CAI. The strategy that was developed from these considerations is based on several assumptions:

1. Effective performance of non-trivial tasks requires that several different subskills be integrated: i.e., "called up" at appropriate points; and that the performer monitor his own performance. (A descriptive model of how performance might be organized was presented by Rigney (1970).

2. Learning how to perform job-related tasks can be accomplished by practicing job-related tasks in permissive environments which offer "look-back" and "look-ahead" analysis and advice to the student.

3. Adaptation to individual differences can be accomplished by (a) student-controlled options, (b) self-tests which indicate to the student his current level of proficiency, (c) "trials-to-criterion" logic that allows differential practice on difficult and easy parts of the task, and (d) "fall back" levels in which drills in necessary subskills are available. These assumptions are based on an appreciable amount of experience in observing and analyzing the performance of procedural and troubleshooting tasks in electronics maintenance.

Their implications are essentially prescriptive. As Atkinson and Paulson (1972) pointed out, instructional psychology will have to operate

in a prescriptive mode for some time without substantial guidance from learning theory until appropriate theory can be developed.

SECTION V. COMPUTER PROGRAMS FOR PERFORMANCE TRAINING

Introduction

Several intermediate models of the current programs were developed, with each succeeding model incorporating improvements relating either to more efficient internal processing or to additional instructional features. Several different programming languages were used at different times, as a matter of convenience or cost-reduction. The initial logic was written in LISP 1.85, and was oriented toward troubleshooting basic circuits and operating test equipment (Rigney and Towne, 1970). It was tested on students in a Naval electronics school with a data module composed of list-structures, describing a symptom-malfunction matrix for a phase-shift oscillator and the task structure for determining the frequency and voltage of an unknown signal with an oscilloscope.

LISP is an interesting language, with powerful list-processing capabilities. Its recursive features are particularly appropriate for processing task structures. Since it was possible to write special recursive functions and to call them in a program, the initial development of the basic characteristics of the TASKTEACH programs was greatly facilitated by the use of this language. An example of a typical LISP function is given in Appendix A. The LISP version of TASKTEACH was capable of administering practice in using test equipment, or a drill in identifying front-panel controls and their modes of operation, or practice in troubleshooting a basic circuit. The student could shift

back and forth from one of these three to any other one. It was anticipated that these features would be useful in automated laboratory training.

The availability of LISP in the beginning of this work was a fortunate circumstance. LISP is, however, not necessarily a good language to use for implementation; it is relatively slow, it is not widely available, it does not do computations efficiently, and it is a difficult language to learn. It also became economically more appropriate to use different, more conventional programming languages. Consequently, subsequent development of the computer programs was done using commonly available programming languages. The current versions are in Fortran IV, because it is currently the best language available on the cheapest time-sharing system available. Differences among programming languages like Basic, PL/1 and Fortran IV are not so great that translation from one to another is a major problem. Each has minor advantages and disadvantages for a particular kind of application.

TASKTEACH Programs

Currently, the generic term, TASKTEACH, includes two large computer programs, each composed of a number of subroutines. Both programs are developments from work originally done with the LISP language. However, both programs contain new features and both necessarily use logic that is different from the logic that would be typical of LISP programming. Although these two programs were developed separately, they could be readily used together by the addition of a small executive routine. Both programs are organized in a similar way, which probably is a standard way to organize fairly large programs for core-swapping and to preserve

a certain amount of open-endedness. The general format is illustrated in Figure 1.

Both programs incorporate the instructional strategy discussed above, although they implement it differently. Both are what Uttal (1968) calls generative, in that they generate or "put together" the interaction with each student. Therefore the content of this interaction is unique to a student. The programs do this by, in effect, "solving the problem" or "performing the task" along with the student. Since the programs do it right, they "know" when the student makes an error, or is using an ineffective strategy, and they "know" what advice to give a particular student when he asks for one of several categories of advice provided for in the programs. With this approach, a great deal of information processing goes on between a student's input--an action or a command--to the programs and the programs' replies to the student. We believe this is an appropriate use of computer power in the long term. Two or three seconds is sufficient time for a computer to accomplish a rather startling number of information-processing operations, at whatever level they may be viewed, considering that cycle times today range from around 300 nanoseconds to around 2 microseconds, and only one to three cycles may be used to execute a machine-language instruction. In the short term, this is a relatively expensive approach to use on some current kinds of data-processing systems. It is not particularly expensive, however, if used in a dedicated system based on minicomputers.

The two programs included in TASKTEACH are described separately below. The troubleshooting program was designed to give students who are attending a school and are learning "theory" by other means, the opportunity to practice troubleshooting in a permissive and supportive

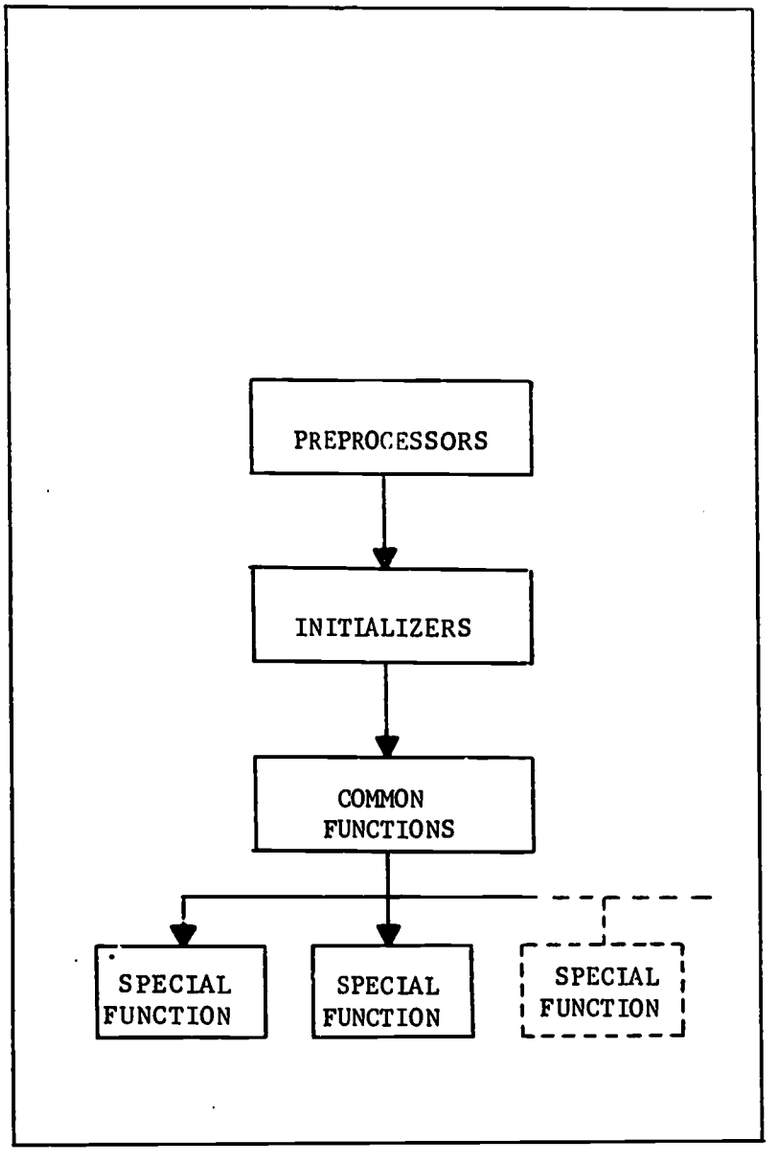

Fig. 1. General modular structure of programs.

learning environment. The procedural program was designed to teach students who might or might not be attending a school on the same equipment, how to operate that equipment. Students who know nothing about the equipment would necessarily have to spend more time on identifying and locating controls and displays than those who had some familiarity with it. The program was designed to work with either population.

We emphasize, again, that our concern up to this point has been with developing software, the terminals we used to test these programs are not desirable for student use. Although the troubleshooting program could be used effectively with a simple alphanumeric CRT or a teletype, the procedural program should not be used for students with such terminals. Its potentialities can be exploited only with a graphics terminal with a pointing device for input.

SECTION VI. COMPUTER-AIDED PERFORMANCE TRAINING IN DIAGNOSTIC SKILLS

This program operates on a data module descriptive of essential features of a particular device to simulate equipment characteristics and states, to generate the interaction with the student, including providing on-demand advice to him, and to record student responses. The program is modularized, as is shown in Figure 2, modules consisting of subroutines that can be called from other modules, or by commands the student can use.

The two modules called MAIN and ACTON must be resident in core while the program is running. The other modules can be read into core from disk memory as needed. These are concerned with implementing specific functions needed only at particular times. Following a brief discussion of these subroutines, interactions with the student generated by the subroutines that do communicate with the student are illustrated in Figure 3. Instructions for using the program are reproduced in Appendix B.

Troubleshooting Program Subroutines

MAPPER. This subroutine may be used to preprocess a data module to reduce CPU costs when the student is on-line. In effect, it re-arranges certain parts of the data into arrays at a level intermediate between the human analyst's output and the processing required during interaction with the student. MAPPER also processes the data module's disk file to compute pointers to use in accessing parts of the file. Since it is a utility program, it is not shown in Figure 2.

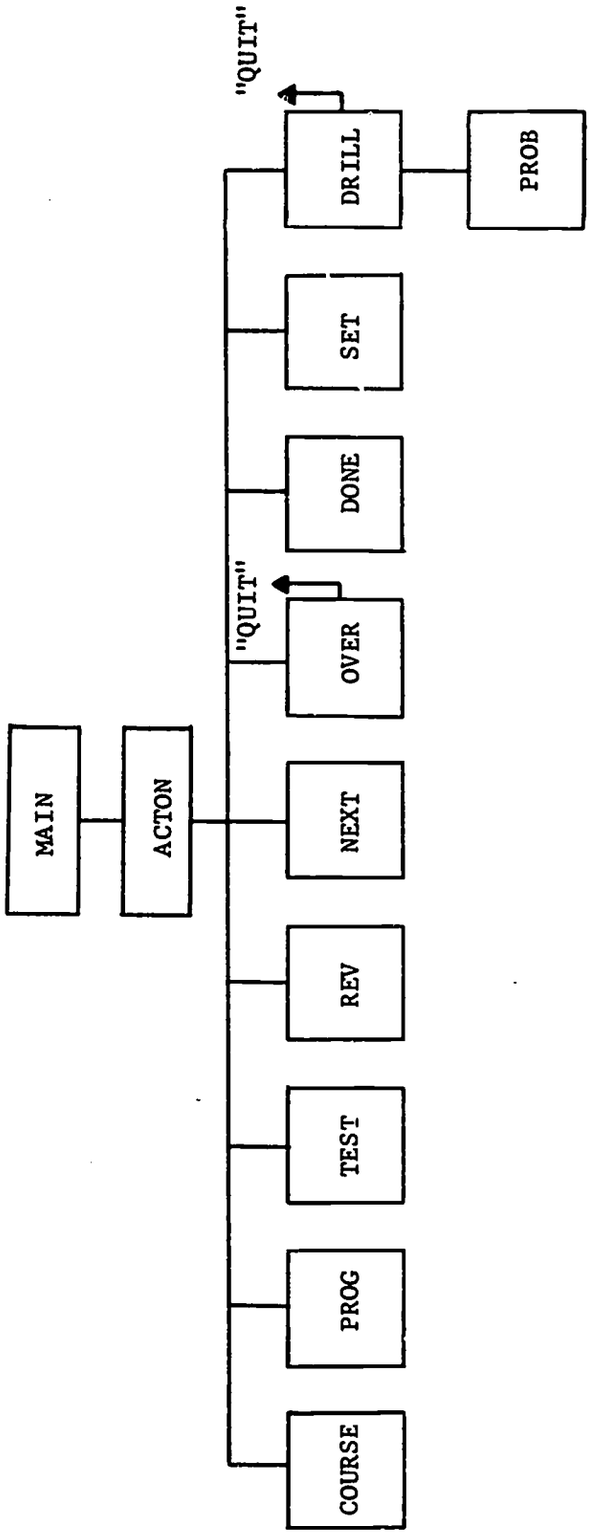

Fig. 2. Diagnostic (troubleshooting) program modules.

ACTION(S)? PROG

THERE ARE (52) POSSIBLE MALFUNCTIONS REMAINING

ACTION(S)? I1,I2

29. [VIDEO SWEEP] IS (NORMAL)

30. [CURSOR SWEEP] IS (NORMAL)

ACTION(S)? PROG

THERE ARE (49) POSSIBLE MALFUNCTIONS REMAINING

ACTION(S)? S2-2,I16

31. [NORM-ALIGN SWITCH S2] NOW SET TO (ALIGN)

32. [ALIGN VIDEO BEAM] IS (NORMAL)

ACTION(S)? S2-1,I16

33. [NORM-ALIGN SWITCH S2] NOW SET TO (NORM)

(---> [ALIGN VIDEO BEAM] SHOULD NOT BE CHECKED IN THIS CONFIGURATION)

ACTION(S)? S2-1

(---> [NORM-ALIGN SWITCH S2] ALREADY SET TO (NORM))

ACTION(S)? PROG

THERE ARE (48) POSSIBLE MALFUNCTIONS REMAINING

ACTION(S)? REV

ENTER THE MALFUNCTION(S) WHICH YOU SUSPECT? M51

[A5C9 SHORTED; OR K8B OPEN CONTACT 5] (COULDN'T BE THE MALFUNCTION)

CHECK YOUR STEP NUMBER (1)

(FOR SOME REASON, YOU ARE NOT SUSPECTING THE ACTUAL MALFUNCTION)
THERE ARE (48) OTHER MALFUNCTIONS WHICH COULD CAUSE THE INDICATIONS YOU HAVE RECEIVED

LEGEND:

( ) Computed or retrieved from data module based upon prior performance.

[ ] Retrieved from data module--prior performance does not affect.

___ Underline--student input.

Remaining text printed from fixed formats in program.

Fig. 3. Typical dialogue between student and troubleshooting program.

MAIN. MAIN establishes initial conditions with the subject; subject number, course number, problem number, etc. It also contains logic for decoding student inputs and for calling different subroutines required by different student inputs. In this sense it is a simple executive routine.

ACTON. This subroutine is responsible among other things for keeping the simulation of equipment states updated; e.g., front panel settings and malfunction possibilities not yet eliminated, and for checking the validity of proposed student actions against these current states. It is the most essential, the longest, and the most complex subroutine in the program. ACTON must be resident in core, with MAIN, while the program is running with a student. However, ACTON does not communicate directly with the student.

COURSE. This short subroutine primarily reads records from disk files into core, preparatory to administering a particular course to a student.

PROG. Progress is one of the on-demand advisory functions the student can use by typing in the command, PROG. This subroutine then looks at the current state of the equipment that the student is troubleshooting and informs the student of the number of defined possibilities for malfunctions remaining (or the number eliminated) up to that point. The communication with the student is illustrated in the instructions to the subject.

TEST. The instructional strategy allows the student to take a self-test any time he wishes, or to take an end test when he thinks he is ready. Two small subroutines, STES and TEST, actually are responsible for managing the necessary processing. Both are embedded in MAIN. The student communicates his wishes by using the commands STES and TEST.

REV. This subroutine manages the processing for the command, REV, which the student can use to ask the program's advice about possible causes of the symptoms he has observed up to that point in a trouble-shooting problem. The subroutine must look at the current state of the problem and at the individual student's action history to provide this advice.

NEXT. This student can ask for advice about the next step to take during troubleshooting, by entering the command, NEXT. Various considerations could determine the fault-area reduction strategy used in the subroutine. Currently, the subroutine uses a simplified Bayesian, essentially a half-split, algorithm. The student can, as in all other cases, take the advice of the program or not. If he does not know how to make the test suggested as the next step, he may ask for a list of controls that must be set for the test.

DONE. This subroutine does the necessary processing after the student indicates to the program, by entering DONE, that he thinks he has solved the problem. If it is a practice session, the program will give the student knowledge of results. If it is a test session, the program will go on to the next problem.

SET. This small subroutine is necessary only when the program is used without front panel simulators, or the actual equipment, and the student cannot check the current state of the front panel control settings visually. Typing in the command, SET, will result in a list of controls and their positions that were last changed.

OVER. This subroutine, and its corresponding command, allows the student to review a problem after he has finished. OVER provides for a selective review, since it is likely that the student would be curious only about some particular aspect of his performance.

DRILL. The corresponding command, DRILL, calls this subroutine, which does the processing for administering two types of drills. The student can ask for a drill in how to make front panel tests, by entering I, or for a "backwards troubleshooting" drill if he enters M. These drills are only two of many that could be conceived and implemented to help students bring themselves up to the integrated performance level.

PROB. This performs operations associated with problem selection, and is called from MAIN and DRILL. This subroutine is called automatically when the student starts each problem.

The instructions in the use of this program are given in Appendix B.

Some typical dialogue between computer and student is shown in Figure 3. This interaction has been marked to illustrate the source of various sections of text. Underlined words were entered by the student. Output in parenthesis was either retrieved from the data module or computed; in either case the particular test depended entirely upon the malfunction being assumed by the program and the student's prior steps. Text enclosed in boxes was retrieved from the data module, and does not depend upon the malfunction or the student's prior work.

SECTION VII. COMPUTER-ASSISTED PERFORMANCE TRAINING IN PROCEDURAL SKILLS

This program evolved from the less complete logic initially incorporated in the LISP program. That logic worked quite well in tests with students at a Naval electronics school. Task structures could be defined efficiently as list structures. Recursive functions were written for the purpose operated on these lists. When all programs comprising TASKTEACH were converted to a less esoteric language, this part of the logic was expanded to include more clustering operators and to provide three instructional modes: Instruct, Practice, and Test. Additional commands were added for the student's use during learning, and subroutines were written to present auxiliary information in the form of photographic slides and "to dig deeper" verbal information.

Program Subroutines

The current program includes the subroutines shown in Figure 4.

DATRED. This is a preprocessor subroutine that performs checks for errors in an ASCII data module prepared by a task analyst, substitutes internally used codes for certain types of verbal material, creates informative lists about the data, and writes the data module and lists on a binary random disk file, if no errors were found. If the analyst did make errors, this subroutine will describe to him the error(s) it has identified. It will ask the analyst to correct the errors and to rerun the subroutine.

CAPTIVE. This is the executive subroutine. It initializes conditions, starts communications with the student, reads into core from appropriate

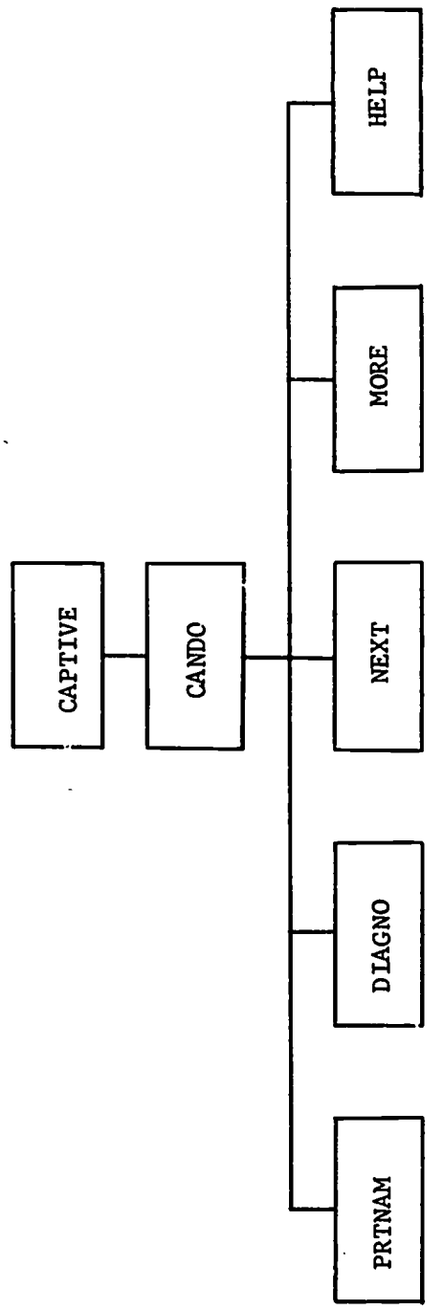

Fig. 4. Procedural program modules.

course files, gets the student started out on the course, accepts the system commands the student uses to communicate with the program, and calls appropriate subroutines.

CANDO. This does the basic operations on the data module to generate interaction with the student. It decodes clustering operators and performs the different processing operations required by each. It keeps track of the student's progress in the task structure and communicates with the student.

PRTNAM. This short subroutine translates between the internal world numeric codes of the program and the external world verbalization required by the student, by reading goal and action names from a file and printing them on a terminal.

DIAGNO. DIAGNO gives a verbal description of how to accomplish goals. It is called when the student types a "D" followed by a goal number.

NEXT. This subroutine is called by the command, NEXT, it gives the student the next action(s) or goal(s) that could logically be performed.

MORE. This is the subroutine that, when called by MORE, reads appropriate "to-dig'deeper" information about a goal or an action from a random access disk file, and displays this information to the student. It also operates a random access slide projector to show the student appropriate slides.

HELP. This subroutine is called by the command, HELP, if the student gets lost in a task structure, he may use this command to be told where he is and be shown a map of the task structure.

System Commands for Interaction and Control

The program for procedural training provides three "modes" of interaction with the student; Instruct, Practice, and Test. Unlike the program described earlier, this program makes no assumptions about the previous experience of the student. It can be used with completely naive students so far as the subject-matter is concerned. The Instruct mode will lead the student by the hand, action-by-action and goal-by-goal through a task structure, giving him "to-dig-deeper" information and showing him slides along the way. If the student wishes, he can explore an entire task structure, without attempting to practice performing it, or, he can, after being instructed in each goal-action cluster, practice recalling it.

The Practice mode allows the student to practice learning a task structure in any of several ways. He can go through the "top level," learning the sequence of major goals first, and then practice lower level goal-action clusters, or he can take the opposite approach, or he can select particular segments of the structure for concentrated practice. The student can ask that a particular cluster be repeated over and over as he is learning its elements.

The Test mode allows the student to "try out his wings," as it were, by trying to perform the task structure without any help. The various system commands the student can use to control the program are described below, in relation to the modes in which they are available. A brief description of each of these is given below.

1. MAIN. This allows the student to change the goal he wants to work on.

2. MODE. The student can change to another of the three modes by using this command.

3. W#. The student must tell the program the goal for which he wishes to learn the actions. W stands for work, # is the goal number.

4. D#. Stands for Describe (list) the actions associated with goal #.

5. P#. Indicates the action the student has performed.

6. GO. Is used by the student in the practice mode to clear the CRT screen when he wishes to practice recalling actions.

7. R#. Means repeat the action cluster associated with goal #.

8. S#. Asks the program to show Slide #.

9. CR. (Carriage return) is used at the end of every student input to signal the computer that the input is ended. When used after S#, it will cause the next of a set of slides to be displayed if there is more than one slide in a set. If not, CR will cause the "standard" slide to be displayed.

10. NEXT. Asks the program, in Mode 1, to list the next action or goal to be performed.

11. HELP. Asks the program to tell the student where he is in a task structure and to show him a map of the structure.

12. BREAK. Is a time sharing system command that can be used to stop the program.

13. QUIT. Tells the program the student wishes to terminate a session.

All of these commands are described in greater detail in the instructions given in Appendix C. Figure 5 summarizes the availability of different system commands in the three different modes. Figure 6

COMMANDS	MODE 1 INSTRUCT	MODE 2 PRACTICE	MODE 3 TEST
MAIN			
MODE			
W#			
D#			
P#			
GO			
R#			
S#			
CR			
NEXT			
HELP			
BREAK			
QUIT			

Available

Not available

Fig. 5. Availability of commands in modes.

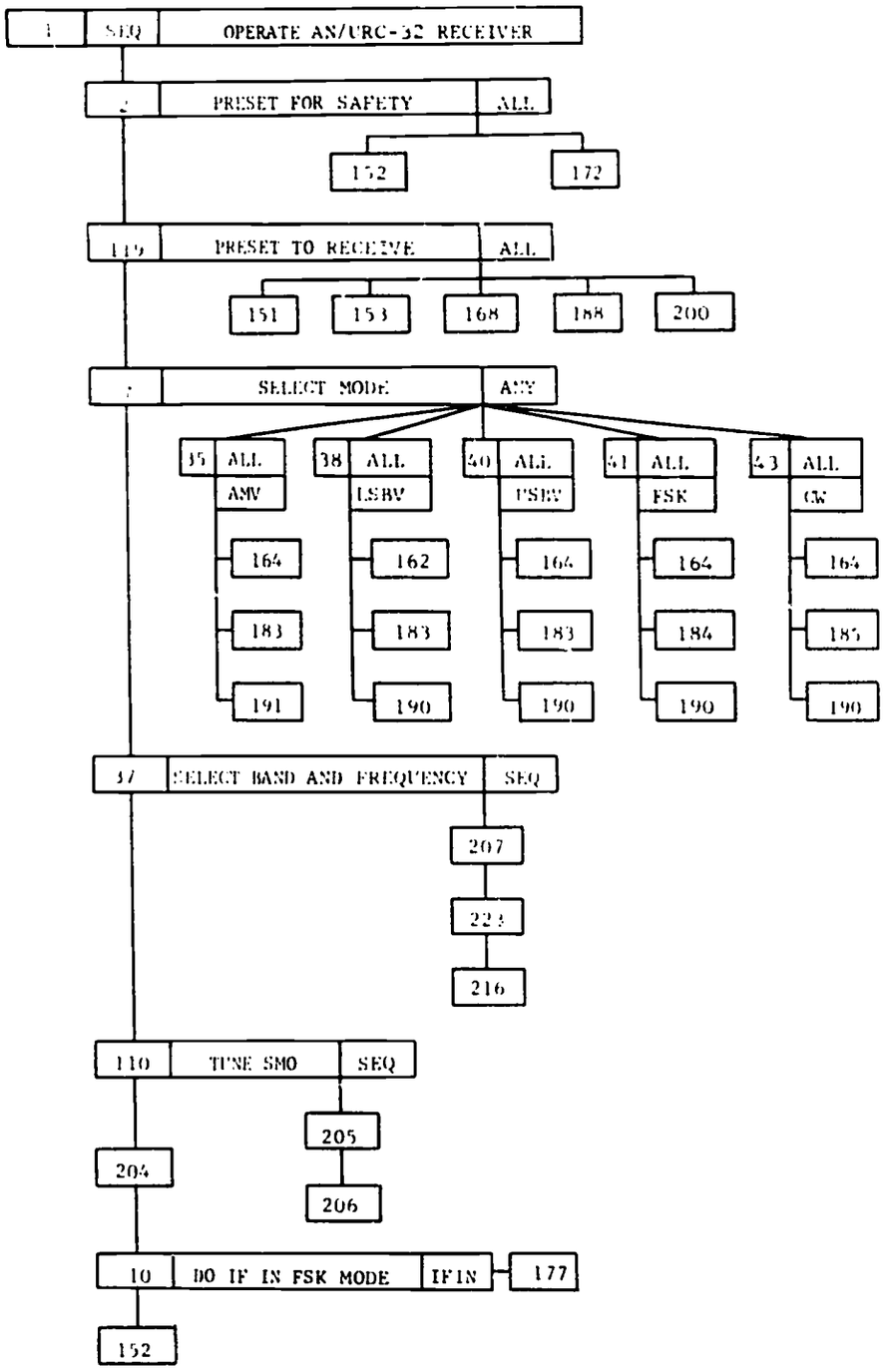

Fig. 6. Graphic representation of a task structure. (Numbers refer to actions performed on the front panel)

illustrates one way of diagramming a task structure. Figure 7 presents a short example of a typical dialogue between a student and the program.

```
>?P207,P223,P216

 "SET BANDCHANGE SWITCH (4S1) TO SELECTED BAND"
 "SET DIAL LOCK (4S3) UP (RELEASE)"
 "SET FREQUENCY CHANGE CONTROL (4FC) TO A FREQUENCY"

 YOU HAVE NOW COMPLETED
 (-"SELECT A BAND AND A FREQUENCY")

>?NEXT

 NEXT YOU MAY PERFORM

 (-"TUNE SMO")

>?W110,P205,P204

 -"TUNE SMO"
  "HOLD OPERATE/TUNE SWITCH (4S2) TO TUNE FOR 1-2 SECONDS"

 ("ADJUST RECEIVER GAIN POT (5R15) UNTIL AUDIO IS HEARD"
  ISN'T PART OF THE GOAL ON WHICH YOU ARE WORKING)
  LAST INPUT ACCEPTED WAS (P205)

>?P206

  "RELEASE OPERATE/TUNE SWITCH (4S2)"

 YOU HAVE NOW COMPLETED
 (-"TUNE SMO")

>?P204

  "ADJUST RECEIVER GAIN POT (5R15) UNTIL AUDIO IS HEARD"

>?W10

 -"ADJUST BFO IF IN FSK RECEIVE MODE"

 -"ADJUST BFO IF IN FSK RECEIVE MODE"
 REQUIRES NO ACTION BECAUSE
 (-"SET CONTROLS FOR FSK")
 HAS NOT BEEN DONE

>?P152

  "SET POWER SWITCH (11S1) TO OFF"
  YOU HAVE COMPLETED THE MAIN GOAL
```

LEGEND:

- ( ) Computed or retrieved from data module based upon prior performance.
- ▭ Retrieved from data module--prior performance does not affect.
- ___ Underline--student input.
- Remaining text printed from fixed formats in program.

Fig. 7. Typical dialogue between student and procedural program.

SECTION VIII. PROGRAM TESTING

It was possible to do some limited testing of both troubleshooting and procedural programs. The objectives were to assess how well the programs performed with students without the presence of the actual equipment, to discover errors in program logic and data modules, to observe student acceptance, and to collect data regarding the use of the different options in the programs. This program testing is an essential first step that must precede evaluation of the programs in a training environment. This latter evaluation involves many, sometimes conflicting, considerations (Bond, 1970) and should be done over a sufficiently long period of time to allow for a series of interrelated studies to be done with reasonably large samples of subjects. Cost is an important consideration, here. It is not economically feasible to do this with a commercial time-sharing system. The two programs that have been described here are being converted to run in a new type of programmable terminal that will be an economical device to use as a "stand alone CAI system." This device is described in a subsequent section.

Troubleshooting Program Tests

The original LISP program was tried out at the Naval Schools Command, Treasure Island, with data modules for practice in troubleshooting a phase-shift oscillator, and practice in determining the frequency and peak voltage of a sine wave with an oscilloscope. For troubleshooting practice, students were given a multimeter and a phase-shift oscillator

encased in an aluminum box. Test points were brought out to the surface of the box, in the form of plug-in jacks. The student could take DC or AC voltage or resistance measures at any of these jacks. Malfunctions were inserted by the instructor via two rows of 15 toggle switches each, one row to open and the other to short components. There were two variations in the program with regard to student and program roles in information-sampling operations. The program could be set to accept the student's readings from a multimeter (with adjustable tolerances), or to give the student a numerical value for DC, AC, or ohms at a test point. In this case, the student's ability to use the multimeter and its accuracy were excluded from the loop. In both cases, the student was asked by the program to judge whether the value was high, normal, or low. The student proceeded, using the program options then available (PROG, NEXT, HIST) until he believed he had located the malfunction. He then entered MAL and the number of a shorted or open component he suspected.

Practice in ascertaining the voltage and frequency of an unknown wave form required a signal generator, probes and leads, and an oscilloscope. The student had to operate the oscilloscope, informing the program, via a teletype terminal, of his actions. He in turn was given knowledge of results, and advice on-demand, and was told what slides to view by operating a random-access slide projector manually.

Enough students were run on the LISP program to ascertain that it worked and that students were interested enough to use it. It also was clear that more sophisticated terminals were necessary for teaching procedural skills.

The troubleshooting program subsequently was developed to give students practice in troubleshooting from front-panel symptoms. This

required entirely new logic to allow the program to "track" the student in making front panel control settings, and to do other processing required for simulating this level of equipment complexity.

This program was tested with a small sample of students from a Naval electronics school. Most of these students were in the last stage of the school on the equipment they were going to practice troubleshooting. It was anticipated that these students would be able to comment on the validity and realism of the simulation of this equipment, the AN/SPA-66.

The students were run two at a time for a week's session. They were given an introduction to the program, shown the reference manuals supplementing the technical manual that had been prepared for off-line study, and were allowed to practice troubleshooting on-line with the program for the remainder of the week until Friday morning. On that morning, they were given an on-line troubleshooting test. The students were able to put in about three and one-half days of troubleshooting practice. They were left alone, to do as they pleased after the first morning's instruction and familiarization session. Without exception, they chose to practice on-line from 8 a.m. to 5 p.m., with a few breaks and time out for lunch. Although a monitor was standing by, the system was essentially automatic. The test for the first few students consisted of ten problems, ten more problems were added later to increase the difficulty of the test. The program automatically recorded eleven items of information about each student as he was working. These were written on one large disk file. The ten students who participated generated 42,896 characters of data on this file.

A program, LINKT, was written to process the student file. This program "deloused" the file, converted time in seconds to minutes, and

analyzed the data by student or by problem. The analysis by student could be over all problems or over subsets of problems of specified size.

A summary of these data is given in Table 1.

The entries are means per problem, except for N (number of problems attempted in a practice or test session). The categories in the table are:

Time: Time in minutes taken per problem; i.e., to find a malfunction. The students were not given knowledge of results during the test.

Tests: The number of front panel tests made during a problem.

Repl: The number of times the student asked the program to "substitute" a good component (actually, fault area) for a suspected one.

Solved: The proportion of problems attempted that were solved.

PROG, REV, OVER, SET: The average number of times each of these advice commands was used by each student.

FPER: The average number of front panel errors made per problem by each student.

N: The total number of problems attempted by each student.

The front panel test drill and the "backwards troubleshooting" drill were not used at all by some students, quite a bit by others. This also was true of the self-test, these frequencies are shown on the next page.

Table 1

Data from Test of Troubleshooting Program

PRACTICE SESSION

STUDENT NO.	TIME	TESTS	REPL	SOLVED	PROG	REV	OVER	SET	FPER	N
1	9.14	10.53	1.58	.64	.30	.22	.00	.05	.00	74
2	16.39	16.42	1.06	.81	.56	.19	.14	.42	.00	36
3	14.22	17.45	1.18	.90	2.50	.98	.15	.33	.00	60
4	20.67	15.36	1.11	.80	.91	.76	.09	.64	.00	45
5	26.54	22.57	1.25	.89	.32	.39	.04	.32	.46	28
6	14.56	16.29	1.31	.94	.81	.48	.04	.13	.23	48
7	15.25	19.34	1.49	.86	.17	.12	.03	.08	.27	59
8	14.19	15.83	1.44	.79	.33	.67	.06	.35	.90	48
9	16.75	19.33	1.10	.85	2.23	.15	.03	.08	1.03	40
10	18.29	18.92	1.14	.86	1.31	.57	.31	.57	.49	51

TEST SESSION

STUDENT NO.	TIME	TESTS	REPL	SOLVED	SET	FPER	N
1	12.10	16.40	3.40	1.00	.00	.00	10
2	8.10	16.60	1.00	1.00	.10	.00	10
3	13.33	21.33	1.00	1.00	.22	.00	9
4	14.25	23.75	1.00	1.00	.50	.00	4
5	15.25	18.20	1.45	.80	.00	.25	20
6	12.75	21.30	1.15	.95	.00	.30	20
7	12.65	22.20	1.35	.90	.00	.40	20
8	12.96	16.58	2.04	.50	.17	.42	24
9	18.25	21.55	1.90	.90	.10	.10	20
10	15.80	18.80	1.50	.85	.00	.00	20

Table 2

Frequency of Use of Self-Tests and Drills
During Practice Sessions

Student No.	STEST	IDRILL	MDRILL
1	0	0	0
2	3	0	4
3	1	2	22
4	0	0	8
5	4	0	1
6	14	6	7
7	8	1	0
8	7	5	4
9	11	0	0
10	6	0	0

Students were interviewed at the end of the week, after taking the test. They were asked the following questions. "Majority"and "minority" comments are summarized under each question.

1. What was the best feature of TASKTEACH?

Maj: It took you through all the variations of front panel settings and front panel tests as they are used for troubleshooting, and related these to the circuits. It forced you to think about these relationships. It made you look "deeper" for front panel symptoms than you do in the regular troubleshooting practice.

Min: Its quickness--you could cover a lot of problems in a short time.

2. What was the worst feature of TASKTEACH?

Maj: You couldn't see the (AN/SPA-66) CRT.

Min: Block diagrams (in reference manual) for some sections were not supplied.

3. Which of the troubleshooting support functions was most useful to you?

Maj: PROG and REV (invariably discussed together).

Min: None.

4. Which was least useful to you?

Maj: SET

Min: OVER (at this time, OVER was not a selective review--it listed the student's entire history)

5. Did you use the (front panel test (I) and "backwards troubleshooting (M)) drills?

Maj: M drill was used more, was considered more helpful.

Min: Several did not use either drill.

6a. What improvements would you suggest in the program?

Maj: (Two suggestions) make OVER a selective review and present CRT symptoms graphically rather than verbally.

Min: None

6b. What improvements would you suggest in the references?

Maj: None. They were far better than what they had at the schools.

Min: Front Panel Tests and Front Panel Organization manuals not necessary if you knew something about the SPA-66.

7. At what point did you feel you understood you knew how to use TASKTEACH?

Maj: Tuesday morning (first day of practice)

Min: None

8. Do you believe something like TASKTEACH would be a worthwhile addition to the Navy schools?

(Unanimous): Yes. Some amplifying comments were:

I learned more about the SPA-66 in one week than I did in three weeks at the school.

Worthwhile--it exposed me to more possible troubles.

Procedural Program Tests

The procedural program was tested using students from a local trade-technical school. The objectives were to test the program logic, to identify errors, and to look for major differences between using the program with the actual equipment front-panel present for the student to work on and using colored slides of the equipment front-panel. The full potential of this program will not be realized until it can be converted to run in a graphics terminal with a light pen, or with front-panel simulators, to reduce the level of verbalization in the student-program interaction. Therefore, only a few subject (six) were run to identify major problems, if any. As soon as it was apparent the students accepted the program, could learn how to perform the procedural tasks, and major trends in usage patterns were identified, data collection was stopped.

The students, none of whom had ever seen the AN/URC-32, the equipment they were to learn to operate, were from electronic communications classes in a technical school. They were divided into two random groups. One group used the actual equipment and the other used the photographic slides during their on-line practice from the program. All students were instructed in how to use the terminal (alphanumeric CRT operating at 300 baud and random access slide projector under program control) and in using the features and options in the program. The students then proceeded at their own individual rates. The task was to learn the procedures for, first, putting the AN/URC-32 into any of five receive modes; and, second, to set the AN/URC-32 up for transmitting in any of five modes. In some respects, this was a fairly rigorous test of the

effectiveness and appeal of the system, since the AN/URC-32 surely must be one of the most poorly human-engineered devices in the Navy (controls are scattered haphazardly over eleven different units and are misleadingly or poorly labeled) and these students had no intrinsic reasons for wanting to learn how to operate this device. They probably would never see it again.

Students worked at the terminals until they considered themselves ready for the performance test. They could take a self-test to determine their readiness by using the test mode (mode 3) of the program. The performance test consisted of putting the AN/URC-32 into a receive and then into a transmit mode (selected by the monitor) with the power on. They used the alphanumeric terminal with the program in test mode during this test. During sessions, the program recorded actions by the students, times, and errors, on disk files.

The slide group made a total of eight errors during the performance test and the front panel group made a total of two errors. All of these were either type 8, attempting to perform an action not part of a goal, or type 11, attempting to perform an action out of sequence.

Times to learn the tasks were inflated by time-sharing system down time and time-sharing system response lags, which were appreciable during the program test. It was necessary, in fact, to change to a different sub-system. Thus, time as a measure of "learning rate" is not reliable. However, mode 1 was used most, mode 3 next, and mode 2 least. The fact that the students did use mode 3, the self-test mode, rather heavily, is of some interest. Also, the slide group required an average of eleven elapsed hours on-line versus seventeen for the front panel group. Usage

trends and errors were highly correlated between the two groups, although the group learning from slides used the HELP function slightly more during practice, and used the NEXT function somewhat less. Error patterns also were highly correlated; both groups made the same kinds of errors. The most frequent type of error was attempting to perform an action that was not part of the goal on which the student was currently working.

Table 3 and 4 presented below give frequencies of usage of different system commands and frequencies of different types of errors, as described on the left hand column of the table, for both groups combined.

Table 3

Mean Frequency of Use of System Commands by Students
N=6

	MODE 1		MODE 2		MODE 3	
	Course 1	Course 2	Course 1	Course 2	Course 1	Course 2
NEXT	15.33	44.66	3.83	.16	.99	.16
MAIN	18.49	28.50	7.33	31.00	14.16	26.16
HELP	3.49	4.33	3.33	.16	.33	.16
MODE	9.16	7.16	5.83	3.16	4.66	5.99
R	0	.33	.16	.33	0	0
D	39.33	91.99	1.16	.16	.33	.49
P	167.49	374.33	63.16	136.32	203.66	435.83
M	1.83	.66	.16	0	0	0
W	75.49	291.66	5.99	4.50	79.16	314.33
S	18.83	43.50	2.33	5.50	.83	8.33

Table 4

Mean Frequencies of Errors
N=6

ERRORS	MODE 1		MODE 2		MODE 3	
	Course 1	Course 2	Course 1	Course 2	Course 1	Course 2
1	2.49	8.49	2.33	2.83	3.99	3.33
2	.50	.16	0	0	0	0
3	2.49	1.16	.16	0	1.16	.16
4	0	0	3.99	.16	.99	.16
5	0	0	1.66	.16	.33	.49
6	0	0	0	0	0	0
7	0	0	0	0	.33	.16
8	19.49	86.66	6.99	6.83	10.16	37.50
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	7.16	7.83	2.83	2.49	3.33	5.83
12	4.16	5.83	2.99	1.16	.99	1.49
13	0	0	0	0	0	0
14	0	.16	0	0	0	0
15	0	0	0	0	0	0
16	4.66	5.61	1.33	2	.49	1.16
17	0	.99	0	0	0	0
18	0	.16	.33	0	0	0
19	0	.33	0	.16	0	0
20	0	0	0	0	.16	.83

1 There is no action or goal #XX (for DXX, PXX, MXX, or WXX)
2 XX is not a goal # for DXX
3 XX is not a goal # for WXX
4 NEXT in modes 2 and 3
5 D in modes 2 and 3
6 M in mode 3
7 HELP in mode 3
8 XX not part of goal on which working
9 XX not required because of answer to IFEX question
10 XX wrong path on IFIN
11 Out of sequence
12 Already done
13 Wrong path on IFIC
14 UNDO action not same as corresponding ANY action
15 For PXX where XX is goal #, difficulty level of XX is greater than current level
16 Main goal already done (PXX, WXX)
17 Break key in response to UNDO question
18 Invalid slide number (SXX where ≤ XX > 88)
19 R in mode 1 or 3
20 No slide of map in mode 3

SECTION IX. FUTURE DEVELOPMENT

Some of the possible configurations of hardware and software for CAI systems were discussed in earlier work (Rigney, Fromer, and Teplitzky, 1968). They range from large networks with one or more large computers to small, stand-alone systems based on minicomputers. The latter have some advantages for use in certain types of environments. For example, where the subject-matter is classified and there are not sufficient local resources to sustain a larger, time-sharing configuration with central CPU and peripheral terminals. We doubt that "either-or" arguments for these two extremes are appropriate. The variables to consider are complex and there probably is room in the world for several different configurations. A small programmable graphics terminal recently became available on the market for use as a stand-alone CAI system. This device is relatively inexpensive, yet it is surprisingly powerful if its inherent capabilities are recognized and are fully exploited. It also could be a forerunner of even more powerful and less expensive programmable graphics terminals made possible by the remarkably swift advances in integrated circuit, microprogramming, and memory technologies.

The programmable graphics terminal includes two integral minicomputers, 12K of core, and an I/O interface for controlling peripherals. It will be the basis for a stand-alone system providing randomly accessed photographic slides under program control, static and animated graphics, simulated front-panels with controls and displays under program control, and both light-pen and keyboard inputs. Each of these different features

contributes uniquely to the training capabilities of the system. The light-pen provides for pointing responses and allows a reduction in the verbal interaction between student and program. The student can respond to and interact with visual imagery. The graphics feature provides for visual imagery, especially imagery that changes in relation to student responses, and for real-time animation. The random-access slide projector, with large back-projection screen, provides the color, definition, resolution and realism of photographic slides that is impossible to obtain by any other comparably inexpensive means. The on-line front-panel simulators provide for realistic simulation of equipment front panels and of tasks performed on those interfaces. The programmable keyboard provides a capability for displaying any special graphics symbols that are needed, since the keys call display subroutines, and these can be changed to suit the application. This feature also allows the creation of "function" keys. These can be used by the student to control events in the instructional environment. We see then, that this small system does provide the user with a remarkably powerful potential likely to be limited more by the users' imagination than by anything else.

The TASKTEACH programs are being converted to run in these programmable graphics terminals. System analysis indicates that the essential features of the troubleshooting and of the procedural programs can be implemented in assembly programs using less than 12K words of core, including a data-module for the program to operate on. In an austere configuration, different data modules, representing different "courses," can be loaded from magnetic tape casettes. Practicing procedural tasks, e.g., the operation of equipment, can be immensely facilitated by substituting pointing responses with a light pen for verbal responses made via a keyboard.

APPENDIX A

EXAMPLE OF LISP 1.85 FUNCTION

Some of the "flavor" of the LISP 1.85 language is illustrated by a small function, called BEST, written for the purpose of finding the ordinal position of the smallest value in a list of sums of squares, called POWER. This particular function was used in the LISP version of the troubleshooting program to provide the information for selecting the best next test to recommend to the student. That test which would most closely approximate splitting the remaining hypothesis set in half could be selected by counting the hypotheses in each of the two subsets that would be produced by applying each possible test. The number of elements in each subset was squared and the two squares were added. The smallest sum-of-squares would identify the desired test. It was necessary to know the ordinal position of this test in the set of possible tests to use in the NEXT function in the program. The BEST function is listed below:

```
(BEST
  (LAMBDA (POWER)
    (COND
      ((NULL (CDR POWER))
        1)
      ((LESSP (CAR POWER)
          (CAR (NTH (CDR POWER)
               (SETQ K (BEST (CDR POWER))))))
        1)
      (T (ADD1 K)))))
```

A few notes may be of use to the reader who wishes to gain some feeling for how a recursive LISP function operates. This is not to say that these notes will please a hardened Lisper, or will be an easy guide

to understanding LISP. In the above listing, BEST is the name of the function. LAMBDA is used to define functions with arguments that are to be evaluated. POWER is the name of the argument to be evaluated, in this case a list of integers representing sums of squares, each of two integers squared. COND is the conditional function of LISP. It takes an indefinite number of arguments, called clauses. For the purposes of this illustration, the first expression of a clause will be labelled p_i, the following expression in the clause will be labelled e_i. These expressions are considered in sequence. The expression, p_i, is evaluated and its value is classified as true (equal to T), or false (equal to NIL). If the first expression is true, then e_1 is evaluated, and the value of the conditional is the value of e_1. NULL has the form null(x) which is equivalent to not(x). CDR is a primitive function in LISP which separates the tail of a list from the first element. The CDR of 12 3 7 6 8 2 9 would be 3 7 6 8 2 9. CAR is a primitive function that returns the first element of a list. The CAR of the above list would be 12. LESSP is an arithmetic function of the general form lessp(x;y). It is true if $x < y$; NIL (false) otherwise. NTH has the general form nth(x;n). Of these arguments, x is a list, and n is a positive integer. The value of NTH is the list whose first element is the nth element of list x. In the above function, x is (CDR POWER) and the value of n is found from evaluating (SETQ K (BEST (CDR POWER))). SETQ is a LISP function of the general form set (x;y). It sets x (without evaluating it) to the value of y. (Here, it sets K to the value of (BEST (CDR POWER)). T is a way of setting p_i to be always true, so that the following e_i is always done. ADD1, add1(x), adds 1 to x, in the above example, to K. We now rewrite

the function with p_i and e_i labelled, and with the elements of NTH, that is, x and n marked:

```
(BEST
  (LAMBDA (POWER)
    (COND
p1    ((NULL (CDR POWER))
  e1    1)
p2    ((LESSP (CAR POWER)
          (CAR NTH* (CDR POWER) (SETQ K (BEST (CDR POWER)))))
  e2    1)                  x              n
p3    (T)                                                  *nth [x,n]
  e3      (ADD1 K)))))
```

In looking for the ordinal position of the lowest value in a list of sums-of-squares called POWER, say, (12 3 7 6 8 2 9), we find that the following will be true:

p_1: If the list (POWER) has only one element, then the position of the BEST test is 1.

p_2: Else, if the first element (CAR POWER) is less than the BEST of rest of the list (BEST (CDR POWER)), then the position of the BEST test is 1.

p_3: Else, the position of the BEST test is 1 plus the position of the BEST test in the rest of the list (ADD1 K).

In a list, (12 3 7 6 8 2 9), the function will find the 6th element in the list, which is 2. If, on the other hand, the order was (2 12 3 7 6 8 9), the function would find 2 to be the first element. The value of BEST, in this case, would be 1 instead of 6.

It is worthwhile noting that the original LISP program consisted of 34 special functions of this general nature, most of them considerably more complex than BEST.

APPENDIX B

INSTRUCTIONS FOR USING THE TROUBLESHOOTING PROGRAM

A New Tool for the Learner to Use

The first industrial revolution substituted machines for manual labor. In the second industrial revolution, computers perform and control operations that formerly had to be done directly by human intelligence. Computers store human intelligence in the form of computer programs. In this way, operations can be done at a time or place distant from the human originator, they can be done over and over, and they can be done at great speeds.

The computer that is the heart of the time-sharing system you will use has a basic cycle time measured in billionths of a second. Although you will be on-line with this computer for an hour or more per session, it will require only a few seconds of processing time for all of its transactions with you, and it will be serving many other customers during that time.

(TASKTEACH) is the computer program you will be using on this time-sharing system. You will use it by operating a CRT terminal with a keyboard like a typewriter. This program was designed to serve you, the learner, as a powerful tool you can use to help you learn how to troubleshoot a device, such as a radar repeater, or an automobile, using symptoms available from "indicators" built into the device. You will have control over how you use this tool. It can give you practice in making "front

panel tests;" in "backwards troubleshooting" in which you select a particular malfunction and try to predict the symptoms it would produce; and in "forward troubleshooting" in which you attempt to localize an unknown malfunction by using symptoms available from built-in indicators. You can choose to do any one of these drills anytime you wish, by giving the program a short command. When you wish to test your proficiency in troubleshooting, you can ask the program to administer a self-test to you. On the basis of the results of the self-test, you may decide that you need to review certain topics, or that you need more practice. After more review of practice, you should take a self-test again. When you decide, on the basis of self-tests, that you have mastered a particular block of problems, you are ready to take a test for the record. In this case, you can ask the program to give you a test, which will be like the self-test, except that the results will be recorded and will be available to your instructor.

If you decide you want to review certain topics, you can log off the time-sharing system and refer to the reference material that is provided. It is likely that you will find it convenient to cycle through several on-line and off-line sessions before you take a test for the record.

When you take a self-test or a test for the record, you should use only the Technical Manual with the block numbers (fault area numbers) on the schematics and block diagrams. Do not use the other references when taking these tests. When you reach the fleet, you will have to rely on the Technical Manual.

The TASKTEACH program contains several features you can use to assist you in learning while you are engaged in on-line drills. You can select any one of these by giving the program a short command word. Each of these commands and what it does for you is described on the next page.

The Equipment Reference Manuals

When you are off-line and wish to read about some topic concerning the device you are learning to troubleshoot, you should start with the reference manuals that are provided. These have been organized to present useful information at two levels of detail; for quick reference and to dig deeper. They also are indexed to the TECHNICAL MANUAL, so that you can quickly find additional information and familiar schematics and block diagrams there. There are four of these Equipment Reference Manuals:

Front Panel Controls and Indicators. In the quick reference section of this manual, all these controls and indicators are illustrated, labelled with proper designations, and described in terms of their functions. The controls and indicators are grouped according to nine categories to assist you in remembering them.

In the to-dig-deeper section, drills in the operation of the equipment, using these controls and indicators, are provided. To use this section, you should have abailable the AN/SPA-66 and the Range Rings Calibrator. The drills in this section will tell you how to set up the equipment and will lead you through a detailed lesson on the use of each control.

Front Panel Tests. This manual gives you an overview of the basic front panel tests, and a detailed description of how to perform them. You can use this as a guide while practicing these tests on the equipment itself, and as a reference source for reading about front panel tests after you have been practicing on the terminal.

Signal Flow Paths. Eleven block diagrams constitute the quick reference section of the manual. These illustrate the major signal paths

for video and cursor sweeps, electron beam intensity control, off-centering inputs and controls, electron beam focus control, range strobe yards and miles counters, and timing section. Blocks in these diagrams are numbered. The numbers are indexed to the TECHNICAL MANUAL by Appendix C3 in this manual.

The to-dig-deeper section of this manual contains a description of each block in a signal flow path and what it does. This will give you an overall understanding of the organization of the circuits in the AN/SPA-66 from the important viewpoint of how they affect front panel indicators, primarily the intensified elements that appear on the CRT. To go into even more detail, you can find a description of each numbered block in the diagrams, in terms of inputs, conversion circuitry, outputs, special conditions, and page numbers in the TECHNICAL MANUAL where you will find diagrams, schematics and further discussion.

Fault Area Summaries. In this manual, you will find a discussion of each fault area you can isolate by making front panel tests and interpreting the symptoms you see on front panel indicators. You probably will refer to this manual after a troubleshooting session on-line with the time-sharing terminal.

The fault area numbers in this manual correspond to the fault area numbers used in the data base for the TASKTEACH program. For each fault area, one or more examples are given of a component failure that could result in the front panel symptoms you observed.

To dig deeper, use the reference data for each fault area, reproduced after the examples of malfunctions, to find diagrams and text in the TECHNICAL MANUAL.

Troubleshooting Advice

Finding out why a system is not functioning properly is required of many different specialists, from physicians to plumbers. People, automobiles, television sets, radar repeaters, washing machines, and moon rockets break down all too frequently. Diagnosing them requires a set of skills common to all of these specialists, plus a great deal of knowledge specific to the system. General steps in diagnosis are:

Listen to the Customer's Complaint. He may or may not know what is wrong. You will have to check what he says, but listen to him and ask questions to draw out all the possibly useful information he has.

Check the State of the System. To do this, you will have to use information-sampling procedures. Very often these involve special instruments and tests which are expensive and difficult to perform. Therefore, it is smart to get all the information you can as simply as you can, first. The physician uses direct observation of the patient, listens with a stethoscope, takes temperature, pulse, and blood pressure before he orders expensive laboratory tests. You should use the information quickly available to you on front panel indicators to diagnose the state of the equipment. The pattern of symptoms you can observe there usually contains sufficient information to reduce the possible sources to a very small percentage of the total circuitry.

The point is, someone has to pay for using information-sampling procedures to get information about the state of a system. You have to pay in terms of time and work done, since unlike the physician, you cannot call on a corps of assistants and pass on the costs to the patient. The customer who wants to use the device pays in terms of lost usage. So, whenever it is possible to do so, get all the low-cost information before you resort to more expensive procedures. Fortunately, front

panel tests can be made quickly and easily. With them, you buy a lot of good information at relatively low cost.

Related to the cost of sampling information is the power of the information you get from using a sampling procedure. Some symptoms tell you very little, others pinpoint the malfunction exactly. Obviously, a high-cost procedure which yields only low-power information is to be avoided. In troubleshooting, powerful information allows you to reduce the fault area rapidly, by eliminating a relatively large proportion of the circuits from the fault area. Weak information leaves you with a fault area not much smaller than it was before you acquired the information.

In most cases, it is necessary to accumulate information by making several tests. A pattern of symptoms will tell a more complete and more accurate story than one symptom alone.

Recognize Symptoms. How do you know that what you are observing is normal or abnormal? You need to compare what you observe with some standard. You may have a description or picture of normal indications to use as referents. You may have learned what normal should be, so you do not need to refer to printed standards. You may be able to calculate the normal value. Regardless of which of these you use, the basic operation is one of comparison of your observation with some standard. In many cases, you will not be absolutely sure that your decision of normal or abnormal is correct.

Under these circumstances, if you make a number of different observations, you can relate each observation to the remaining ones in terms of content and context.

Content refers to the magnitude of a measurement or observation in relation to what it is reasonable to expect. If B+ is 300 volts and your

measurement of B+ is 3000 volts, you should suspect your procedure or voltmeter. If the content of the observation is unreasonable, you probably have made an error in procedure.

Context refers to the pattern of symptoms up to the current observation. Observations should fit together to make a logical story. Observations that contradict or do not fit that story are suspect and also cast doubt on every thing up to that point. Such an observation should be checked. Make the observation again, watching your procedure carefully. If it holds up, make a confirming check. That is, if the observation is true, then you should predict the outcome of sampling some other related indication. Make the second observation to see if the predicted outcome occurs. If it does, the context you have built up from previous information may be wrong. If it does not, either check the first observation again or ignore it temporarily and go on making other checks, they may resolve the conflicting information.

Relate Symptoms to Causes. Diagnosing depends on knowing how elements of the system are interrelated. An expert on a particular equipment builds up a mental "model" of how the equipment works--and doesn't work. He can relate his observations to this model. Because he knows signal flow paths and other relationships, he can infer probable causes for a pattern of symptoms.

It takes a lot of experience to build up a mental model as detailed as that of the expert's; you have to do a lot of thinking about relationships and a lot of testing your mental model against reality by using it. You will need to look up answers to questions you formulate for yourself in the tech manual or other references. If your mental model lets you

down when you are troubleshooting, you have to find out why and change that part of the model.

The important point is, this kind of learning depends on your, the learner's, operations on available information. When you have done the organizing, relating, reality testing, and reorganizing, you will have a mental model you can remember and you can rely on for relating symptoms to causes.

Reduce the Fault Area. Possibilities for malfunctions that would cause a certain symptom pattern can be considered from a functional and a geographic standpoint. Each circuit in the equipment performs some function and each circuit has a physical location. For example, if the symptoms seem to be due to absence of the negative time-share gate, you then need to look for the location of the circuits that produce this gate.

When you are troubleshooting from front panel indications, the signal flow paths will be guides to circuit chains concerned with major functions, and to the circuits which modify these functions in various ways. There are three major geographic areas in the AN/SPA-66 where these circuits are found: in the (p-c board) subassemblies in the rack just below the operator's writing shelf, in the power supply unit below this rack, and on p-c boards and in electro-mechanical subassemblies behind the front panel.

Of course, when you are troubleshooting from the front panel, you cannot physically bracket the ends of a fault area and reduce it by moving the brackets closer. You have to work with functions which affect the intensified elements you can see on the CRT or the outputs of other indicators. The key relationships to know are those among functions, intensified CRT elements and other front panel indications, and front panel controls. You have to know how to operate front panel controls to

make each intensified element visible, and to make it possible for the effects of those functions which affect each intensified element to be observable.

Starting the Program

In this section, all entries typed by the student are underlined; the remaining words are typed automatically by TASKTEACH. TASKTEACH will ask for your student number, it will list the equipments which it can help you with, and will have you pick a problem.

To begin a session with TASKTEACH, type your student number and select a problem:

EXAMPLE:

```
WHAT IS YOUR STUDENT NUMBER? 26

SPA-66 REPEATER
THERE ARE 136 PROBLEMS

WHICH PROBLEM WILL YOU TAKE? 122
```

Troubleshooting with TASKTEACH

When you start TASKTEACH, you will automatically be in the troubleshooting mode. When you enter a problem number, the program will select that problem, representing a malfunction, unknown to you, which you must find.

A troubleshooting session with TASKTEACH consists of an exchange of information between you and the program. TASKTEACH will print:

```
ACTION(S):?
```

whenever it is awaiting input from you. You may enter indicators you wish to "observe," you may enter the malfunction which you believe is causing

the symptoms you have received, or you may command TASKTEACH to help you in a variety of ways.

After TASKTEACH has provided the information or the help you required, it will print ACTION(S):? again, and wait for you to respond. This interaction continues until you solve the problem, at which time you may choose another, or terminate the session.

Checking Indicators. You will be given a list of indicators for the equipment you are working with. To check the status of an indicator, you type I followed by the indicator number. TASKTEACH will respond with the status of that indicator.

EXAMPLE:

```
ACTION(S):? I3,I5,I6,I7,I8
---> TRACKING STROBE SHOULD NOT BE CHECKED IN
THIS CONFIGURATION.
17.  RANGE STROBE IS NORMAL
18.  RADAR VIDEO IS NORMAL
---> OFF-CENTERING IN RESPONSE TO DRA INPUT SHOULD NOT
BE CHECKED IN THIS CONFIGURATION.
---> OFF-CENTERING IN RESPONSE TO AEW INPUT SHOULD NOT
BE CHECKED IN THIS CONFIGURATION.
```

The example also shows that you may enter two or more indicators you wish to check, by separating them with commas.

Setting Switches. You will be given a list of switches for the equipment you are working with. This list provides the name and code for each switch, and the names of each of its settings. It also indicates the initial setting of each switch. This initial mode is assumed at the start of each problem.

To change a switch setting, type the switch code, followed by a "-" mark and then the setting number.

EXAMPLE:

```
ACTION(S):? S13-1,S2-2,A7S1-1,S16-6
10. RADAR SELECTOR SWITCH S13 NOW SET TO CONVENTIONAL
11. NORM-ALIGN SWITCH S2 NOW SET TO ALIGN
12. SWEEP-BOTH-CURSOR SELECTOR A7S1 NOW SET TO SWEEP
13. RANGE RINGS/MILES SWITCH S16 NOW SET TO 50 MILES
```

As with indicators, you may enter several switch changes on one line, if you separate them with commas.

Replacing Components. You will be given a list of possible malfunctions in the equipment you are working with. When you think you know what is causing the trouble with the equipment, type M followed by the malfunction number which you suspect. You should then make more checks to determine if the abnormal symptoms have been corrected.

EXAMPLE:

```
ACTION(S):? I3
1. INDICATOR THREE IS OFF-CENTER

ACTION(S):? M5
2. M5 IS NOW REPLACED

ACTION(S):? I3
3. INDICATOR THREE IS NORMAL

ACTION(S):? DONE
YOU FOUND THE PROBLEM...M5

WHICH PROBLEM WILL YOU TAKE?
```

This example also shows that you type DONE to tell TASKTEACH that you have corrected the problem. If you find that the problem persists after "replacing" a component or circuit, you should do more trouble-shooting until you can find the actual problem.

Controlling the Program

TASKTEACH is a tool which you can use to substantially reduce the time and effort required to learn about complex tasks.

To make it provide the help you want, you will type certain command words. These commands are explained on the following page.

General Commands.

RUN This starts the program running.

QUIT When you want to terminate a session with TASKTEACH, type QUIT. The terminal will print a "-" mark. You then type BYE. You may then turn off the terminal.

EXAMPLE:

ACTION(S):? QUIT

BYE

Troubleshooting Commands.

1. REV Use this when troubleshooting to ask the program to REView a list of possibilities for malfunctions which you wish to know about. The program will tell you the following about each possibility you listed:

   (1) It is a good possibility--at this point it could be the malfunction.

   (2) You obtained enough information in a preceding step (Step No.) to have eliminated this possibility.

   The program also will tell you the following:

   (1) You are not suspecting the actual malfunction. The program will list possibilities you overlooked.

   or (2) The actual malfunction is among those you suspect.

   The following is an example of the use of REV. Observe that you must identify each possibility you list with M followed by a number, and that this must be followed by a comma to separate it from the next possibility.

EXAMPLE:

ACTION(S)? REV

ENTER THE MALFUNCTION(S) WHICH YOU SUSPECT? M35,M36,M45

S16A-9 OPEN CONTACT COULDN'T BE THE MALFUNCTION
CHECK YOUR STEP NUMBER 2

A6R38 OPEN IS A GOOD POSSIBILITY

A6R93 OPEN; OR C12 SHORTED COULDN'T BE THE MALFUNCTION
CHECK YOUR STEP NUMBER 2

THE ACTUAL MALFUNCTION IS AMONG THOSE WHICH YOU SUSPECTED

2. PROG Use this when troubleshooting if you want to get some idea of the PROGress you have made in localizing the fault area. The command PROG will cause the program to print out the number of remaining possibilities for the malfunction, in terms of blocks defined in the block diagrams you will use, which remain up to this point.

EXAMPLE:

ACTION(S)? PROG
THERE ARE 2 POSSIBLE MALFUNCTIONS REMAINING

ACTION(S)?

3. SET Use this command any time you want to refresh your memory about the settings of the front panel controls. The program will print out the current settings of any controls which are not in their original settings.

EXAMPLE:

ACTION(S)? SET

THE SWITCHES ARE SET IN THEIR INITIAL SETTINGS
EXCEPT FOR THE FOLLOWING:
RANGE RINGS/MILES SWITCH S16 IS SET TO 20 MILES
RADAR SELECTOR SWITCH S13 IS SET TO ELSCAN

4. OVER You would use this command at the end of a troubleshooting problem if you wanted a selective "replay" of your work. The program will go back to the first action you request and print out the step number, what you did at that step, and what the result of your action was. The program will ask you if you have any questions about particular fault areas. If you type YES, the program will ask you to enter the fault areas which you are interested in.

EXAMPLE:

```
WHICH PROBLEM WILL YOU TAKE? OVER

HERE IS A SUMMARY OF YOUR WORK
ENTER THE MALFUNCTIONS YOU WISH TO DISCUSS? M45

MALFUNCTION 45 WAS ELIMINATED BY STEP 2
RANGE RING ELEMENT IS ABNORMAL
ENTER THE MALFUNCTIONS YOU WISH TO DISCUSS? 0
ENTER THE STEPS YOU WISH TO DISCUSS? 2,4

2. RANGE RING ELEMENT IS ABNORMAL
THIS ELIMINATED 110 POSSIBLE FAULT AREAS
WOULD YOU LIKE A LIST OF THESE MALFUNCTION? NO

4. RANGE RING ELEMENT IS ABNORMAL
THIS DID NOT ELIMINATE ANY OF THE POSSIBLE FAULTS

ENTER THE STEPS YOU WISH TO DISCUSS? QUIT
```

NEXT Command. Use this command if you want advice about the next test to make during troubleshooting. The program will tell you an efficient next test to make, and will give you a list of front panel controls that are important for making this test. It will not, however, tell you the positions to which these switches must be set.

EXAMPLE:

```
ACTION(S)? NEXT
IF YOU CHECK THE RANGE RING ELEMENT IN THE APPROPRIATE
SWITCH CONFIGURATION(S), YOU WILL OBTAIN USEFUL INFO
DO YOU WANT A LIST OF SWITCHES THAT ARE IMPORTANT? YES
THE FOLLOWING SWITCHES ARE IMPORTANT FOR THIS INDICATOR
RANGE RINGS/MILES SWITCH S16
RADAR SELECTOR SWITCH S13
SWEEP-BOTH-CURSOR SELECTOR A7S1
NORM/ALIGN SWITCH S2

ACTION(S)? S13-2
8. RADAR SELECTOR SWITCH S13 NOW SET TO ELSCAN

ACTION(S)? I4
9. RANGE RING ELEMENT IS NORMAL
```

DONE Command. Use this command while troubleshooting to tell the program you have finished a problem. This will normally occur after you have found what you think is the malfunction, the program has "substituted" a new component, and you have checked the appropriate front panel indicators again and see that the abnormal symptoms have disappeared.

If you enter DONE, and you have not in fact found the real malfunction, the program will tell you that you did not solve the problem, and it will ask you if you wish to continue this problem. If you reply YES, the program will ask you what actions you want to take next. If you reply NO, the program will ask you for another problem number and you can start on a new problem.

If DONE is used while you are taking a test, the program will select the next problem in the test, without telling you anything else.

EXAMPLE 1:

```
ACTION(S)? DONE

YOU FOUND THE PROBLEM ... A6R38 OPEN
```

EXAMPLE 2: (during test)

```
ACTION(S)? DONE
PROBLEM 2
ACTION(S)? I1,I2,S16-5,I4

1. VIDEO SWEEP IS NORMAL
2. CURSOR SWEEP IS NORMAL
3. RANGE RINGS/MILES SWITCH S16 NOW SET TO 20 MILES
4. RANGE RING ELEMENT IS NORMAL

ACTION(S)? DONE
PROBLEM 3
```

Drill Command. This command allows you to practice making front panel tests, or doing "backwards troubleshooting." When given this command, the

program will ask you what you would like to discuss. If you enter an I followed by a number, it will give you a drill in checking that front panel indicator. If you enter an M followed by a number it will ask you to list the front panel indicators which you believe would be affected by that malfunction. The program will inform you if your selections were correct, remind you of other indicators you overlooked, and, if necessary, review with you the particular switch settings necessary for making the effects of the selected fault area observable on a selected indicator.

EXAMPLE 1:

WHICH PROBLEM WILL YOU TAKE? DRILL

WHAT WOULD YOU LIKE TO DISCUSS? M36

ENTER THE INDICATORS WHICH YOU BELIEVE
WOULD BE AFFECTED BY M36? I3,I4,I5

TRACKING STROBE IS NOT AFFECTED BY M36 IN ANY MODE

YOU ARE RIGHT ABOUT RANGE RING ELEMENT
FOR THIS MALFUNCTION TO AFFECT
RANGE RING ELEMENT

TO WHAT POSITION WOULD YOU SET RANGE RINGS/MILES SWITCH S16? 4

NO, THIS CONTROL MUST BE SET TO POSITION
5: 20 MILES

WITH THESE SETTINGS, RANGE RING ELEMENT IS ABNORMAL

RANGE STROBE IS NOT AFFECTED BY M36 IN ANY MODE

EXAMPLE 2:

WHAT WOULD YOU LIKE TO DISCUSS? I7

IN ORDER TO CHECK OFF-CENTERING IN RESPONSE TO DRA INPUT
TO WHAT POSITION WOULD YOU SET RADAR SELECTOR SWITCH S13? 2

RIGHT ... ELSCAN

TO WHAT POSITION WOULD YOU SET OFF-CENTERING SWITCH S14? 2

NO, THIS CONTROL MUST BE SET TO POSITION
1: OFF CENTER

TO WHAT POSITION WOULD YOU SET MAN-DRA SWITCH S3? 2

RIGHT ... MAN-DRA

WITH THESE SETTINGS, OFF-CENTERING IN RESPONSE TO DRA INPUT MAY BE CHECKED.

Test Commands.

STES Use this command when you want to give yourself a SELF-TEST. The program will administer the test to you.

EXAMPLE:

WHICH PROBLEM WILL YOU TAKE? STES

CHOOSE A PROBLEM FOR SELF-TESTING? 122

ACTION(S)? S16-5,I4

1. RANGE RINGS/MILES SWITCH S16 NOW SET TO 20 MILES
2. RANGE RING ELEMENT IS ABNORMAL

ACTION(S)? M36

5. A6R38 OPEN IS NOW REPLACED

ACTION(S)? DONE

YOU FOUND THE PROBLEM ... A6R38 OPEN
SELF-TEST IS OVER.

TEST When you are ready to take a TEST for the record, use this command. The program will administer the test. The results will be available to your instructor.

EXAMPLE:

WHICH PROBLEM WILL YOU TAKE? TEST

PROBLEM 1

ACTION(S)? I1,I2,I3,I4

1. VIDEO SWEEP IS NORMAL
2. CURSOR SWEEP IS NORMAL
---> TRACKING STROBE SHOULD NOT BE CHECKED IN THIS CONFIGURATION
---> RANGE RING ELEMENT SHOULD NOT BE CHECKED IN THIS CONFIGURATION

ACTION(S)? S16-5,I4

3. RANGE RINGS/MILES SWITCH S16 NOW SET TO 20 MILES
4. RANGE RING ELEMENT IS ABSENT

ACTION(S)? PROG

WHAT?

Typing Errors

Using TASKTEACH requires a minimum of typing; your inputs are always short combinations of characters. You will, however, occasionally make typing errors. You will seldom suffer from these errors, however, for two reasons:

1. If what you type doesn't make any sense at all, TASKTEACH will know you made a typing error, and will simply ask for the entry again.

EXAMPLE:

YOU TYPE	INSTEAD OF
J4	I4
T15-2	S15-2
OXER	OVER
QUIL	QUIT

2. If what you type makes sense, TASKTEACH will respond. However, this will never take much of your time.

EXAMPLE:

YOU TYPE	INSTEAD OF	RESPONSE
I4	I5	You get indication of I4
M3	M13	You replace malfunction M3
DONE	QUIT	You will terminate current problem; you may quit next if you wish.

During tests, you could be penalized slightly for making a typing error which makes sense to TASKTEACH. For example, if you type I4 instead of I5, TASKTEACH will give you the indication at I4, and it will count that as a reading. This is not very significant, however.

Or, typing M3 instead of M5, during a test, will "cost-you" one extra replacement. This is slightly more serious.

In any case, if you make a typing error, and notice it before typing the carriage return key, you can correct the entry by typing - (Shift O) for each wrong character, and continuing.

EXAMPLE:

WHICH PROBLEM WILL YOU TAKE? STES

CHOOSE A PROBLEM FOR SELF-TESTING? 122

ACTION(S)? S16-I4 ←← 5,I4,S13-2,I4

Practice Session

You whould try using TASKTEACH on sample problems. In this first session, be sure you try out every option in the program.

You may not need to use all of these options, but you do need to know that they are available, and what each one does. The best way to find this out is to explore each, using sample problems at the beginning. Otherwise you may overlook some very useful options.

Also, while familiarizing yourself with TASKTEACH options, you should use the diagrams in the Signal Flow Path Manual, and observe the relationship between these and the special diagrams and schematics in the Technical Manual. Observe that the block numbers in the troubleshooting diagrams are also found in the special diagrams and schematics in the Technical Manual, where these numbers appear inside diamonds with arrows pointing to the output of a possible fault area.

You will be required to use only the Technical Manual when taking self tests, intermediate tests, and the final test for the record. Therefore, during the week you are working with TASKTEACH, you should learn to transition from using the TASKTEACH Manuals to using the Technical Manual.

APPENDIX C

INSTRUCTIONS FOR USING THE PROCEDURAL PROGRAM

This system is designed to assist you in learning complicated procedures, called task structures, such as those required to repair and operate equipment. The program contains a number of features you can use to help you learn. These will be described in detail later. First, a word or two about what task structures are, and about effective ways to learn them.

All human work can be described in terms of goals, actions and conditions for performance of the actions. There is some goal to be accomplished by one or more actions, which must be performed correctly. The correct ways to perform actions may be determined by a variety of conditions, some specific to the kind of work, others more general to all work. There generally is some overall, or final, goal that can be attained only by first accomplishing a series of subgoals, one at a time. The structure that results from describing work in this way can be represented in a "map" that shows goals, actions, and conditions. Such a map shows you the detailed organization of a task structure, and therefore, is useful when you are learning the structure.

The CAI system you will work with makes learning more convenient by assembling information you need, showing you colored slides of devices you are learning about, allowing you to proceed at your own pace, and helping you to correct errors.

The system provides you with control commands that will let you tailor-make your own learning experiences, by using particular combinations of these commands. The system cannot, however, learn for you. Learning is a mostly consequence of the operations you perform on the material to be learned. Several steps are involved:

1. Learn what this CAI system has to offer by reading about its features in the following paragraphs, and then trying all of them out on the example provided, so you know how to use the system. The system provides an Instruct Mode (1), a Practice Mode (2), and a Self-Test Mode (3). You can use Mode 1 to find out all about a task structure, then use Modes 2 and 3 to practice learning it.

2. Learn the series of goals and actions that make up the "backbone" of a task structure. The CAI system will help you do this.

3. Learn the actions that go with each goal by working on one goal at a time in Mode 2. In this mode, you can easily practice the actions required to accomplish a goal. If you need to have the system review them for you, you can ask it to do this with a command called R.

4. Go through one "thread" of the entire task structure, from start to finish, staying in Mode 3 as much as you can. When you can stay in Mode 3 all the way through without making any errors, you are close to mastery. When you can go through in Mode 3 a second time without errors (and without returning to Mode 1 or 2 for help) you have learned that thread sufficiently well to move on.

5. The operations you perform on the material to be learned are of two kinds: (1) interacting with the terminal, where you first learn about the task structure and then try to recall it; and (2) thinking about what

you are learning "--running it over in your mind." This is called rehearsal. During rehearsal, there are many mental operations you can perform to organize the material you are learning, to relate it to what you already know and to check for yourself how much you know. These operations are the key to rapid learning and long-term retention. When you can run through an entire task structure in your mind, recalling the sequence of goals, the actions that go with each, and the conditions for their performance, you will not need the CAI system anymore! The features of the CAI system and the commands you can use to control it are described in the following paragraphs.

1. As pointed out above, there are three modes, starting with Mode 1, which is used primarily to get a description of the procedures to be learned, and progressing to Mode 3, in which you can test yourself on how well you have learned the steps in the procedures. Mode 2 is an intermediate mode that gives you the opportunity to practice recalling the task structure, with the assistance of the system.

Use "MODE" to tell the program you want to change modes. The program then will ask you to enter the new mode number. You can change modes any time you wish. Generally, you would progress from Mode 1, to 2, and finally to 3, as you shift from needing instruction about, to practicing recall of, the structure you are learning. But, if you find Mode 3 too hard the first time, you can go back to Mode 2 or 1. Any time you enter Mode 2, it will give you a list of the major goals in the task structure.

The > ? output from the system indicates it is expecting a command from you. If it prints only a ?, it is expecting you to answer the preceding question with a number (usually) or with a YES or NO.

Statements in quotes preceded by a "-" are goal descriptions. No leading "-" indicates an action description.

In all the following examples, your command or response to the computer is underlined.

EXAMPLE:

>?MODE

NEW MODE?2

-"OPERATE THE AN/URC-32 AS A RECEIVER "
MAY BE ACCOMPLISHED BY PERFORMING
THE FOLLOWING IN THE GIVEN ORDER:

-"SAFETY PRESET CONTROLS "

-"PRESET CONTROLS TO RECEIVE "

-"SELECT A MODE "

-"SELECT A BAND AND A FREQUENCY "

-"TUNE SMO"

"ADJUST RECEIVER GAIN POT (5R15) UNTIL AUDIO IS HEARD"

-"ADJUST BFO IF IN FSK RECEIVE MODE "

"SET POWER SWITCH (11S1) TO OFF "

2. There are six single letter commands: M, D, R, S, W and P. (These are easily remembered if you observe that they are the first letters in the main words of the phrase "Metropolitan Department of Reprocessed Sewage, Water, and Power!") You can use the first three of these commands, M, D and R, to ask for detailed information about a part of a task structure associated with accomplishing a subgoal and to look at photographic slides. M commands the system to give you "MORE" information about goals

or actions. M also directs the system to operate a slide projector and to show you a photograph of the part of the equipment or device appropriate to the current goal or action. If there is more than one slide to see, the number of slides which follow in sequence will be printed on the CRT. You can see these additional slides by hitting the "CR" key on the terminal. When you have seen all of them, if you press the CR key again, the system will return the slide magazine to a "standby" slide. You can use M in Mode 2 for goals and actions. In Mode 1, you need use M only for actions since it is supplied automatically for goals. M is not available in Mode 3.

EXAMPLE:

```
>?M216

SLIDE NUMBER 72 ( 2 SLIDES)

THE FREQUENCY IS READ IN THE AREA LIGHTED BY THE BAND
CONTROL SETTING; ALIGN THE DIGITS IN THIS WINDOW.
>?  CR

SLIDE NUMBER 73
>? CR

SLIDE NUMBER 88 (STANDBY)
```

The D command can be used in Mode 1 to ask the system to list all the actions required to accomplish a goal and to give you more information about the goal. The system also tells you the way you should perform the actions. In some cases, all the actions must be performed in one particular sequence. In others, all actions must be performed, but sequence is unimportant. In still other cases, only one of a list of actions

need be performed. These are the most common conditions. Less common special conditions will be described later.

EXAMPLES:

TRANSMIT

>?D1

-"TRANSMIT WITH AN/URC-32 "
MAY BE ACCOMPLISHED BY PERFORMING
THE FOLLOWING IN THE GIVEN ORDER:

-"SAFETY PRESET CONTROLS "

-"TUNE TRANSMITTER AT SELECTED FREQUENCY "

-"OPERATE IN A TRANSMIT MODE "

SLIDE NUMBER 01

THE AN/URC-32 RADIO TRANSCEIVER IS REALLY MANY DEVICES IN ONE PACKAGE. TO USE IT, YOU MUST FIRST SET FRONT-PANEL CONTROLS TO SELECT THE PARTICULAR "DEVICE" YOU WANT. IN THIS LESSON YOU WILL LEARN HOW TO SET CONTROLS TO USE THE TRANSMITTER IN A SELECTED MODE AND FREQUENCY.

RECEIVE

>?D2

-"SAFETY PRESET CONTROLS "
MAY BE ACCOMPLISHED BY PERFORMING
ALL OF THE FOLLOWING IN ANY ORDER:

"SET XMIT/REC/CW TEST SWITCH TO REC (6S3)"

"SET POWER SWITCH (11S1) TO OFF "

GENERALLY, AVOID ACTUATING ANY OF THESE SPRING-LOADED SWITCHES OUT OF SEQUENCE AFTER POWER IS ON:

PRESS-TO-TALK (PTT) SWITCH (10S1)
OPERATE-TUNE SWITCH (4S2)
XMIT/REC/CW TEST SWITCH (6S3)
PLATE ON/OFF/KEY SWITCH (3S9)
PLATE NO. 1 SWITCH (3S2)
PLATE NO. 2 SWITCH (3S3)

The third single letter command, R, may be used in Mode 2 to ask the system to repeat a goal and the actions associated with it. This will be useful when you are practicing recalling the cluster of actions required to accomplish the goal. You can do this as many times as necessary to assist you in learning.

The fourth single letter command, S, may be used to review the colored slides for a lesson. You will be given a list of all slide titles and magazine slot numbers. You can see any slide about a particular topic by entering S, followed by the slot number.

EXAMPLE:

> ?S5

SLIDE NUMBER 5

The last two of the six single letter commands, W and P, are used when you want to practice performing the task structure. W followed by a goal number tells the system you want to work on accomplishing that goal. W can be used in any mode, and must always be used before you try to perform any actions, so the system will know what goal you are trying to accomplish. When you are in Mode 2, entering W and a goal number will cause the system to automatically list all the actions required for that goal. The system will not do this in Mode 1 or 3.

In any mode, if you select the wrong next goal to work on, the system will respond with an error message and will wait for you to select a goal that would be correct to work on next. As you will see in a moment, if you are in Modes 1 or 2, you can get help, if you do not know what to do

next. Since no help is available in Mode 3, if you don't know what to do, you will have to retreat to Mode 1 or 2.

EXAMPLE:

```
>?W7

 -"SELECT A MODE    "
MAY BE ACCOMPLISHED BY PERFORMING
ANY ONE OF THE FOLLOWING:

 -"SET CONTROLS FOR AM  "

 -"SET CONTROLS FOR LSB "

 -"SET CONTROLS FOR USB "

 -"SET CONTROLS FOR FSK "

 -"SET CONTROLS FOR CW  "
```

The P command tells the system you already know how to accomplish a goal, or that you propose to perform a particular action next. If you tell the system you already know how to accomplish a goal, it will believe you, and, if the goal was appropriate, allow you to proceed to the next goal. If the goal was not appropriate, it will not let you proceed until the error is corrected by your choosing another, appropriate goal. In the case of proposed actions, the system allows you to go on if the action, or actions, you proposed were appropriate and were proposed in the correct sequence. Otherwise, you will receive an error message and be required to try again.

Any time you use the P command in Mode 2, the CRT screen will be cleared of all information. Thus, you should be prepared to remember action or goal numbers you wish to enter with P. Any time you enter P

and a correct goal or action number, the system will "echo back" the goal or action statement.

The modes in which you can use these six commands are shown below:

	Mode 1	Mode 2	Mode 3
M	Yes*	Yes**	
D	Yes		
R		Yes	
S	Yes	Yes	Yes
W	Yes	Yes	
P	Yes	Yes	Yes

*Use only for actions: more information about goals is given automatically by D.

**Use for either goals or actions.

3. You have several other commands that you can use to select goals to work on, to ask for help if you forget where you are in the task structure, or to ask the system to tell you what to do next. Finally, there is a command to use to tell the system you want to quit. These commands are described as follows:

"MAIN" tells the system you want to change the main goal you want to work on. The system will ask you to enter the new main goal number. You can use this command to select different parts of the task structure to learn about.

EXAMPLE:

>?MAIN

NEW MAIN GOAL?7

"NEXT" is available only in Mode 1 to ask the system to tell you what to do next. (In fact, in Mode 1, you can use "NEXT," "D" and "M" to get a complete description of a new task structure, including all the slides for actions.

EXAMPLE:

```
>?NEXT
NEXT YOU MAY PERFORM
 -"PRESET CONTROLS TO RECEIVE "
>?NEXT
YOU MAY DO ALL OF THE FOLLOWING IN ANY ORDER:
  "SET SIDEBAND SWITCH (7S1) to USB "
  "SET OSC CONTROL SWITCH (6S1) to OFF "
  "SET SSB/AM SWITCH (5S4) to AM "
```

"HELP" can be used in Modes 1 and 2 to ask the system to tell you where you are, and to show you a slide of the part of the task structure you are working on. The diagram on the slide shows you a "map" with goals and actions belonging to the goals, and the conditions under which the actions must be performed.

EXAMPLE:

```
>?HELP
SLIDE NUMBER 81
THERE ARE 6 SLIDES FOR THE MAP
YOU ARE CURRENTLY WORKING ON
 -"SAFETY PRESET CONTROLS  "
THE OTHER GOALS IN PROGRESS ARE
 -"TRANSMIT WITH AN/URC-32 "
```

The light green "backbone" of the map shows the major goals and actions that must be accomplished in sequence, starting from the top. The light yellow structures show secondary goals and actions that must be accomplished in exact sequence. Blue parts of the map show parts of the structure that can be done in any order, so long as all actions are performed. Orange boxes in the map specify choice points: only one of those subsequent parts of the structure that are connected to these "ANY" boxes need be accomplished. Purple boxes signify special conditions. For example, the "UNDO" constraints specifies that if you turned something on, or disassembled it, in a particular way, it has to be turned off, or reassembled, in the reverse order. Another special condition is called "IFIN." This specifies that a goal is to be accomplished or an action is to be performed only if some prior condition has been established. For example, you would do something only if you had earlier selected a particular one of several different ways to operate a device. The program will tell you what to do when you come to any of these and other special conditions.

EXAMPLE (IFIN):

> ?D10

-"ADJUST BFO IF IN FSK RECEIVE MODE"
DEPENDS UPON THE COMPLETION OF

-"SET CONTROLS FOR FSK"
IF IT HAS BEEN DONE, THEN PERFORM

"ADJUST BFO CONTROL (6A1C) TO OBTAIN CLEAR SIGNAL"
IF IT HAS NOT BEEN DONE, NO ACTION IS REQUIRED

THE BEAT FREQUENCY OSCILLATOR (BFO) SUPPLIES A SPECIAL INTERMEDIATE FREQUENCY FOR THE CW-FSK UNIT SO THAT 2125 CPS "SPACE" and 2975 CPS "MARK" AUDIO TONES CAN BE SUPPLIED TO THE FSK CONVERTER TO DRIVE A TELETYPE.

Although it is unlikely you would wish to do so, "COUR" can be used to change courses, e.g., from AN/URC-32 receive to AN/URC-32 transmit.

If you want to stop the system from displaying something on the CRT, hit the "break" key once and the program will stop printing. Then tell the system what you want it to do by using one of the other commands. CAUTION: Do not hit the "break" key twice; you will be immediately disconnected from the time-sharing system. Also, when you are first starting on the system, do not hit the break key before the system prints the first >?. If you accidentally do this, you will have to type "RUN" and the system will start over.

"QUIT" is used when you have finished a session and want to "sign-off" the terminal. The system will tell you the point at which program execution stopped, and wait. You then enter "BYE" and you will be disconnected from the time-sharing system.

EXAMPLE:

```
> ?QUIT
 98

 PROGRAM STOP AT 1490

 USED    1.54 UNITS
BYE
0013.46  CRU    0000.87  TCH     0008.79   KC

OFF AT 14:25PDT 0/6/05/72
```

The following figure (next page) shows you the modes in which these commands can be used:

Command	Mode 1	Mode 2	Mode 3
MAIN	Yes	Yes	Yes
MODE	Yes	Yes	Yes
NEXT	Yes		
HELP	Yes	Yes	
QUIT	Yes	Yes	Yes
BREAK	Yes	Yes	Yes

4. You can correct an accidental typing error by holding the shift key down and hitting back arrow (on the "0" key) once for each erroneous character. Then, retype the message from that point.

5. You will be shown how to log on the system. After logging on, you will be asked for your student number, what course you want to take. what main goal, what mode, and what difficulty level. Always enter 1 for difficulty level, other levels are not currently being used.

REFERENCES

Atkinson, R. C., and Paulson, J. A. An approach to the psychology of instruction, Psych. Bull., 1972, 78(1), 49-61.

Bond, Nicholas A., Jr. and Rigney, J. W. Measurement of training outcomes. Los Angeles: University of Southern California, Behavioral Technology Laboratories, June 1970. (Tech. Rep. 66)

Carbonell, Jaime. Scholar, A New Approach to Computer-Assisted Instruction. Naval Research Reviews, October 1972. Arlington: Office of Naval Research, Department of the Navy.

Feurtzig, W. Automated Instructional Monitors for Complex Operational Tasks. Final Report; Bolt, Beranek and New, Inc., October, 1971.

Rigney, J. W., Fromer, R., and Bond, N. A., Jr. The application of time-sharing technology to training requirements at the Electronics Schools Commands, Mare Island. Los Angeles: University of Southern California, Electronics Personnel Res. Group, December 1967. (Tech. Rep. 56)

Rigney, J. W., Fromer, R., and Teplitzky, F. Requirements for a computer-aided instruction system for the U.S. Naval Schools Command, Mare Island. Los Angeles: Electronics Personnel Res. Group, June 1968. (Tech. Rep. 58)

Rigney, J. W., and Towne, D. M. TASKTEACH: A method for computer-assisted performance training, Human Factors, 1970, 12(3), 285-296.

Rigney, J. W. Maintainability: Psychological factors in the persistency and consistency of design. In K. B. DeGreene (Ed.), Systems Psychology. New York: McGraw-Hill, 1970.

Uttal, W. R. Real-Time Computers: Technique and Applications in the Psychological Sciences, New York: Harper and Row, 1968.

ONR DISTRIBUTION LIST

Navy

4 Dr. Marshall J. Farr
Director, Personnel and Training
Research Programs
Office of Naval Research
Arlington, Virginia 22217

1 Director
ONR Branch Office
495 Summer Street
Boston, Massachusetts 02210

1 Director
ONR Branch Office
1030 East Green Street
Pasadena, California 91101

1 Director
ONR Branch Office
536 South Clark Street
Chicago, Illinois 60605

1 Office of Naval Research
Area Office
1076 Mission Street
San Francisco, California 94103

1 Commander
Operational Test and Evaluation Force
U.S. Naval Base
Norfolk, Virginia 23511

6 Director
Naval Research Laboratory
Code 2627
Washington, D. C. 20390

12 Defense Documentation Center
Cameron Station, Building 5
5010 Duke Street
Alexandria, Virginia 22314

1 Chairman
Behavioral Science Department
Naval Command and Management Division
U.S. Naval Academy
Luce Hall
Annapolis, Maryland 21402

1 Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, Tennessee 38054
ATTN: Dr. G. D. Mayo

1 Chief of Naval Training
Naval Air Station
Pensacola, Florida 32508
ATTN: CAPT Allen E. McMichael

1 Chief
Bureau of Medicine and Surgery
Code 513
Washington, D. C. 20390

1 Commander Naval Air Reserve
Naval Air Station
Glenview, Illinois 60026

1 Commander
Naval Air Systems Command
Navy Department, AIR-413C
Washington, D. C. 20360

1 Commander
Submarine Development Group Two
Fleet Post Office
New York, NY 09501

1 Commanding Officer
Naval Air Technical Training
Center
Jacksonville, Florida 32213

1 Commanding Officer
Naval Personnel and Training
Research Laboratory
San Diego, California 92152

1 Commanding Officer
Service School Command
U.S. Naval Training Center
San Diego, California 92133
ATTN: Code 303

1 Head, Personnel Measurement Staff
Capital Area Personnel Service Office
Ballston Tower #2, Room 1204
801 N. Randolph Street
Arlington, Virginia 22203

1 Program Coordinator
Bureau of Medicine and Surgery (Code 71G)
Department of the Navy
Washington, D. C. 20390

1 Research Director, Code 06
Research and Evaluation Department
U.S. Naval Examining Center
Building 2711 - Green Bay Area
Great Lakes, Illinois 60088
ATTN: C. S. Winiewicz

1 Superintendent
Naval Postgraduate School
Monterey, California 93940
ATTN: Library (Code 2124)

1 Technical Director
Naval Personnel Research and Development Laboratory
Washington Navy Yard
Building 200
Washington, D. C. 20390

1 Technical Director
Personnel Research Division
Bureau of Naval Personnel
Washington, D. C. 20370

1 Technical Library (Pers-11B)
Bureau of Naval Personnel
Department of the Navy
Washington, D. C. 20360

1 Technical Library
Naval Ship Systems Command
National Center
Building 3 Room 3
S-08
Washington, D. C. 20360

1 Technical Reference Library
Naval Medical Research Institute
National Naval Medical Center
Bethesda, Maryland 20014

1 Behavioral Sciences Department
Naval Medical Research Institute
National Naval Medical Center
Bethesda, Maryland 20014

1 COL George Caridakis
Director, Office of Manpower Utilization
Headquarters, Marine Corps (A01H)
MCB
Quantico, Virginia 22134

1 Special Assistant for Research and Studies
OASN (M&RA)
The Pentagon, Room 4E794
Washington, D. C. 20350

1 Mr. George N. Graine
Naval Ship Systems Command
(SHIPS 03H)
Department of the Navy
Washington, D. C. 20360

1 CDR Richard L. Martin, USN
COMFAIRMIRAMAR F-14
NAS Miramar, California 92145

1 Mr. Lee Miller (AIR 413E)
Naval Air Systems Command
5600 Columbia Pike
Falls Church, Virginia 22042

1 Dr. James J. Regan
Code 55
Naval Training Device Center
Orlando, Florida 32813

1 Dr. A. L. Slafkosky
Scientific Advisor (Code Ax)
Commandant of the Marine Corps
Washington, D. C. 20380

1 LCDR Charles J. Theisen, Jr. MSC, USN
CSOT
Naval Air Development Center
Warminster, Pennsylvania 18974

Army

1 Behavioral Sciences Division
Office of Chief of Research and Development
Department of the Army
Washington, D. C. 20310

1 U.S. Army Behavior and Systems Research Laboratory
Rosslyn Commonwealth Building, Room 239
1300 Wilson Boulevard
Arlington, Virginia 22209

1 Director of Research
U.S. Army Armor Human Research Unit
ATTN: Library
Building 2422 Morade Street
Fort Knox, Kentucky 40121

1 COMMANDANT
U.S. Army Adjutant General School
Fort Benjamin Harrison, Indiana 46216
ATTN: ATSAG-EA

1 Commanding Officer
ATTN: LTC Montgomery
USACDC - PASA
Ft. Benjamin Harrison, Indiana 46249

1 Director
Behavioral Sciences Laboratory
U.S. Army Research Institute of Environmental Medicine
Natick, Maryland 01760

1 Commandant
U.S. Army Infantry School
ATTN: ATSIN-H
Fort Benning, Georgia 31905

1 U.S. Army Research Institute
Room 239
Commonwealth Building
1300 Wilson Boulevard
Arlington, Virginia 22209
ATTN: Dr. R. Dusek

1 Mr. Edmund Fuchs
BESRL
Commonwealth Building, Room 239
1320 Wilson Boulevard
Arlington, Virginia 22209

Air Force

1 AFHRL (TR.Dr. G.A. Eckstrand)
Wright-Patterson Air Force Base
Ohio 45433

1 AFHRL (TRT/Dr. Ross L. Morgan)
Wright-Patterson Air Force Base
Ohio 45433

1 AFHRL/MD
701 Prince Street
Room 200
Alexandria, Virginia 22314

1 AFOSR (NL)
1400 Wilson Boulevard
Arlington, Virginia 22209

1 COMMANDANT
USAF School of Aerospace Medicine
ATTN: Aeromedical Libraty (SCL-4)
Brooks AFB, Texas 78235

1 Personnel Research Division
AFHRL
Lackland Air Force Base
San Antonio, Texas 78236

1 Headquarters, U.S. Air Force
Chief, Personnel Research and Analysis Division (AF/DPXY)
Washington, D. C. 20330

1 Research and Analysis Division
AF/DPXYR Room 4C200
Washington, D. C. 20330

1 Headquarters Electronic Systems Division
ATTN: Dr. Sylvia R. Mayer/MCIT
LG Hanscom Field
Bedford, Maryland 01730

1 CAPT Jack Thorpe USAF
Department of Psychology
Bowling Green State University
Bowling Green, Ohio 43403

DOD

1 Mr. William J. Stormer
DOD Computer Institute
Washington Navy Yard
Building 175
Washington, D.C. 20390

1 Mr. Joseph J. Cowan, Chief
Psychological Research Branch (P-1)
U.S. Coast Guard Headquarters
400 Seventh Street, SW
Washington, D.C. 20590

1 Dr. Ralph R. Canter
Director for Manpower Research
Office of Secretary of Defense
The Pentagon, Room 3C980
Washington, D.C. 20310

Other Government

1 Dr. Alvin E. Goins, Chief
Personality and Cognition
Research Section
Behavioral Sciences Research
Branch
National Institute of Mental
Health
5600 Fishers Lane
Rockville, Maryland 20852

1 Dr. Andrew R. Molnar
Computer Innovation in Education
Section
Office of Computing Activities
National Science Foundation
Washington, D.C. 20550

1 Office of Computer Information
Center for Computer Sciences and
Technology
National Bureau of Standards
Washington, D.C. 20234

Miscellaneous

1 Dr. Scarvia Anderson
Executive Director for Special
Development
Educational Testing Service
Princeton, New Jersey 08540

1 Professor John Annett
The Open University
Waltonteale, BLETCHLEY
Bucks, ENGLAND

1 Dr. Richard C. Atkinson
Department of Psychology
Stanford University
Stanford, California 94305

1 Dr. Bernard M. Bass
University of Rochester
Management Research Center
Rochester, New York 14627

1 Professor Mats Bjorkman
University of Umea
Department of Psychology
Radhuseplanaden 2
S-902 47 UMEA/SWEDEN

1 Mr. H. Dean Brown
Stanford Research Institute
333 Ravenswood Avenue
Menlo Park, California 94025

1 Dr. Jaime Carbonell
Bolt Beranek and Newman
50 Moulton Street
Cambridge, Maryland 02138

1 Dr. Kenneth E. Clark
University of Rochester
College of Arts and Sciences
River Campus Station
Rochester, New York 14627

1 ERIC
Processing and Reference Facility
4833 Rugby Avenue
Bethesda, Maryland 20014

1 Dr. Victor Fields
Department of Psychology
Montgomery College
Rockville, Maryland 20850

1 Dr. Robert Glaser
Learning Research and Development
Center
University of Pittsburgh
Pittsburgh, Pennsylvania 15213

1 Dr. Albert S. Glickman
American Institutes for Research
8555 Sixteenth Street
Silver Spring, Maryland 20910

1 Dr. Duncan N. Hansen
Center for Computer-Assisted Instruction
Florida State University
Tallahassee, Florida 32306

1 Human Resources Research Organization
Division #3
Post Office Box 5787
Presidio of Monterey, California 93940

1 Human Resources Research Organization
Division #4, Infantry
Post Office Box 2086
Fort Banning, Georgia 31905

1 Human Resources Research Organization
Division #5, Air Defense
Post Office Box 6057
Fort Bliss, Texas 79916

1 Library
HumRRO Division Number 6
P. O. Box 428
Fort Rucker, Alabama 36360

1 Dr. Lawrence B. Johnson
Lawrence Johnson and Associates, Inc.
2001 "S" Street, NW
Suite 502
Washington, D. C 20009

1 Dr. Roger A. Kaufman
Graduate School of Human Behavior
U.S. International University
8655 E. Pomerada Road
San Diego, California 92124

1 Dr. E. J. McCormick
Department of Psychological Sciences
Purdue University
Lafayette, Indiana 47907

1 Mr. Luigi Petrullo
2431 North Edgewood Street
Arlington, Virginia 22207

1 Dr. Robert D. Pritchard
Assistant Professor of Psychology
Purdue University
Lafayette, Indiana 47907

1 Dr. Diane M. Ramsey-Klee
R-K Research & System Design
3947 Ridgemont Drive
Malibu, California 90265

1 Dr. Leonard L. Rosenbaum, Chairman
Department of Psychology
Montgomery College
Rockville, Maryland 20850

1 Dr. George E. Rowland
Rowland and Company, Inc.
Post Office Box 61
Haddonfield, New Jersey 08033

1 Dr. Benjamin Schneider
Department of Psychology
University of Maryland
College Park, Maryland 20742

1 Dr. Robert J. Seidel
Human Resources Research Organization
300 N. Washington Street
Alexandria, Virginia 22314

1 Dr. Arthur I. Siegel
Applied Psychological Services
Science Center
404 East Lancaster Avenue
Wayne, Pennsylvania 19087

1 Dr. Benton J. Underwood
Department of Psychology
Northwestern University
Evanston, Illinois 60201

1 Dr. David Weiss
University of Minnesota
Department of Psychology
Elliott Hall
Minneapolis, Minnesota 55455

1 Mr. Edmund C. Berkeley
Berkeley Enterprises, Inc.
815 Washington Street
Newtonville, Maryland 02160

1 Mr. Dennis J. Sullivan
Training Department
Hughes Aircraft Co.
P.O. Box 90515
Los Angeles, California 90009

1 Dr. Frank J. Harris
Chief, Training Division
U.S. Army Research Institute
1300 Wilson Boulevard
Arlington, Virginia 22209

1 LCOL Austin W. Kibler, Director
Human Resources Research Office
ARPA
1400 Wilson Boulevard
Arlington, Virginia 22209

1 Dr. Robert R. Mackie
Human Factors Research, Inc.
Santa Barbara Research Park
6780 Cortona Drive
Goleta, California 93017